DYNAMIC MODELLING FOR THE PATH TRACKING CONTROL OF A FOUR-WHEEL INDEPENDENT-DRIVE, FOUR-WHEEL INDEPENDENT-STEER AUTONOMOUS GROUND VEHICLE

A Thesis
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Karl Emerson Klindworth

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Mechanical Engineering

November 2017

Fargo, North Dakota

North Dakota State University
Graduate School

Title

DYNAMIC MODELLING FOR THE PATH TRACKING CONTROL OF A FOUR-WHEEL INDEPENDENT-DRIVE, FOUR-WHEEL INDEPENDENT-STEER AUTONOMOUS GROUND VEHICLE

By

Karl Emerson Klindworth

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Majura Selekwa
Chair

Dr. Annie Tangpong

Dr. Mariusz Ziejewski

Dr. Sumathy Krishnan

Dr. Jacob Glower

Approved:

11/16/17
Date

Dr. Alan Kallmeyer
Department Chair

ABSTRACT

Robots that can be reconfigured to perform more than one task would be to consumers. The Four-Wheel Independent-Drive, Four-Wheel Independent-Steer (4WD4WS) robot is well suited for the role of reconfigurable robot due to its extremely high maneuverability and torque control. However, the nonlinear dynamics in conjunction with complex kinematic constraints make the 4WD4WS structure an extremely difficult control problem. As a result of this many who model the 4WD4WS structure make simplifications that aren't realistic for a reconfigurable consumer robot.

A 4WD4WS robot is kinematically and dynamically modeled using both the front and rear path angles and their respective coordinates. High fidelity equations of motion, for robots of arbitrary width, length, and mass, undergoing arbitrary accelerations at arbitrary steering angles have been created that have the potential to increase the path tracking ability of 4WD4WS systems. Simulations show the model behaves realistically, but needs a controller.

ACKNOWLEDGEMENTS

I would like to thank Dr. Selekwa for all the hard work he has put in helping me with this project. I have worked with him for two and a half years and in that time he has taught me numerous lessons that I know will help me over my career, but also in life. He has become a very good friend of mine. I am hopeful that we may get to work together again in the future.

I would like to thank Dr. Kallmeyer for mentoring me over the years. I have gotten to know him very well through the four classes I have taken from him as well as various social gatherings. It is because of teachers like him that I was excited to come back for my Master’s Degree.

I would like to thank my parents for family for always being there for me. Whenever school got difficult I knew a trip home would give me encouragement to get back at the grind. I would not be who I am today without your encouragement.

I would like to thank Joe Cluett. He spent countless hours programming different subsystems (IMU, GPS, WIFI…) for me while also teaching me how to program in C. While we didn’t get the system going, we set the next group up for success.

I would like to thank all the members of the team that helped me with building the experimental system, including: Tyler Lane, Brady Goenner, Willem Bohrer, Sam, Andrew Schlangen, Nate Peterson, and Andy Narvesen.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

COG ....Center-of-Gravity

DOF...Degree-of-Freedom

ICR ....Instant-Center-of-Rotation

2WD..Two-Wheel Drive

4WD..Four-Wheel Drive

STM ....STM32F407VG Microcontroller

PI...Raspberry PI 3B

LIST OF SYMBOLS

W .... The lateral width of the robot. This is from wheel center to wheel center.

H .... The longitudinal length of the robot. This is from wheel center to wheel center.

X_B .... The global lateral position of the center-of-gravity.

Y_B .... The global longitudinal position of the center-of-gravity.

Ψ_B .... The yaw-orientation of the body with respect to the global frame.

X_F .... The global lateral position of the perpendicular bisector of the front center, between the contact point of the front wheels, of the robot body. Referred to as one of the front path coordinates.

Y_F .... The global longitudinal position of the perpendicular bisector of the front center, between the contact point of the front wheels, of the robot body. Referred to as one of the front path coordinates.

X_R .... The global lateral position of the perpendicular bisector of the rear center, between the contact point of the rear wheels, of the robot body. Referred to as one of the rear path coordinates.

Y_R .... The global longitudinal position of the perpendicular bisector of the rear center, between the contact point of the rear wheels, of the robot body. Referred to as one of the rear path coordinates.

X_1 .... The global lateral position of Wheel-Body-1. Wheel-Body-1 is on the front-left of the main body.

Y_1 .... The global longitudinal position of Wheel-Body-1. Wheel-Body-1 is on the front-left of the main body.

Ψ_1 .... The yaw-orientation of Wheel-Body-1 with respect to the global frame. Wheel-Body-1 is on the front-left of the main body.

X_2 .... The global lateral position of Wheel-Body-2. Wheel-Body-2 is on the front-right of the main body.

Y_2 .... The global longitudinal position of Wheel-Body-2. Wheel-Body-2 is on the front-right of the main body.

Ψ_2 .... The yaw-orientation of Wheel-Body-2 with respect to the global frame. Wheel-Body-2 is on the front-right of the main body.

X_3 .... The global lateral position of Wheel-Body-3. Wheel-Body-3 is on the rear-left of the main body.

Y_3 .... The global longitudinal position of Wheel-Body-3. Wheel-Body-3 is on the rear-left of the main body.

Ψ_3 .... The yaw-orientation of Wheel-Body-3 with respect to the global frame. Wheel-Body-3 is on the rear-left of the main body.

X_4 .... The global lateral position of Wheel-Body-4. Wheel-Body-4 is on the front-right of the main body.

Y_4 .... The global longitudinal position of Wheel-Body-4. Wheel-Body-4 is on the rear-right of the main body.

Ψ_4 .... The yaw-orientation of Wheel-Body-4 with respect to the global frame. Wheel-Body-4 is on the rear-right of the main body.

δ_F .... The angle between a line that runs parallel to the axis of the robot and a line that runs tangent to the path from the location of the front path coordinates.

δ_R .... The angle between a line that runs parallel to the longitudinal axis of the robot and a line that runs tangent to the path from the location of the front path coordinates.

δ_1 .... The angle between a line that runs parallel to the longitudinal axis of the robot and a line that runs parallel to the orientation of Wheel-Body-1.

δ_2 .... The angle between a line that runs parallel to the longitudinal axis of the robot and a line that runs parallel to the orientation of Wheel-Body-2.

δ_3 .... The angle between a line that runs parallel to the longitudinal axis of the robot and a line that runs parallel to the orientation of Wheel-Body-3.

δ_4 .... The angle between a line that runs parallel to the longitudinal axis of the robot and a line that runs parallel to the orientation of Wheel-Body-4.

ρ_F .... The magnitude of the distance between the front path coordinate and the location of the Instant-Center-of-Rotation.

ρ_R .... The magnitude of the distance between the rear path coordinate and the location of the Instant-Center-of-Rotation.

ρ_1 .... The magnitude of the distance between the Wheel-Body-1 and the location of the Instant-Center-of-Rotation.

ρ_2 .... The magnitude of the distance between the Wheel-Body-2 and the location of the Instant-Center-of-Rotation.

ρ_3 .... The magnitude of the distance between the Wheel-Body-3 and the location of the Instant-Center-of-Rotation.

ρ_4 .... The magnitude of the distance between the Wheel-Body-4 and the location of the Instant-Center-of-Rotation.

ω_1 .... The magnitude of the rotational speed of the drive wheel for Wheel-Body-1.

ω_2 .... The magnitude of the rotational speed of the drive wheel for Wheel-Body-2.

ω_3 .... The magnitude of the rotational speed of the drive wheel for Wheel-Body-3.

ω_4 .... The magnitude of the rotational speed of the drive wheel for Wheel-Body-4.

L ....The sum of the kinetic and potential energies of a dynamic system, known as the Lagrangian.

T ....The total kinetic energy of a dynamic system.

V ....The total potential energy of a dynamic system.

T_B ....A kinetic energy component of the chassis.

T_i ....A kinetic energy component of a wheel-body.

m_B ....The mass of the Chassis.

m_i ....The mass of each Wheel-Body.

I_B ....The rotational inertia of the Chassis.

I_w ....The rotational inertia for the drive-wheels of each Wheel-Body.

I_s ....The rotational inertia due to the steering of each Wheel-Body.

1. INTRODUCTION

1.1. Background and Motivation

Starting with early industrial revolution, there has been a steady growth in technological advances towards automation of various tasks in human life. Some of these tasks, though necessary, are extremely undesirable due to either the task itself, or the location in which the task is being performed. A common criteria for identifying an undesirable task is the phrase: dull, dirty, and dangerous. Some tasks that fall within this criteria are: warehouse inventory movement, sewer reconnaissance, and bomb disposal.

The introduction of robots, flexibly programmable machines that can handle a variety of repetitive tasks in a variety of environments, was motivated by the presence of tasks identified as dull, dirty, and dangerous. Although the history of industrial robotics can be traced as back as in the 1930's the first programmable industrial robot is believed to have been commissioned by Unimation in the early 60s, [1]. Most of the robots developed along the line of Unimation were industrial manipulators for performing motion in the 3-D space, and were primarily used in industrial manufacturing. Over the years, applications of robotic systems have expanded to include industries such as nuclear and explosives handling, agriculture, manufacturing, logistics, undersea exploration, and for use as personal robots [2].

As the world continues to enjoy the growth of technological advances throughout the next decade, demands for robot applications will grow accordingly. Ground robotic vehicles is the main field of robotics that is likely to see tremendous demands because it creates the platform on which other special purpose robotic devices can be attached and made to reach the intended objective task. Many types of robots, for example, fire-fighting robots, search and rescue robots, mining robots, and domestic service robots, must be able to move from one place to another.

Therefore, no matter what specific task the robot will be intended to accomplish, it will have to have a mobile platform. While there are a plethora of different methods of locomotion including: air, sea, underwater, and ground; ground robots will see the most growth due to the limited number of constraints compared to aerial robots and the broader amount of tasks compared to sea-based robots. Ground robots can take many forms as well, and while maneuverability requirements may stipulate a certain robot locomotion system, such as legged movement, most of these robotic vehicles will rely on wheels for their locomotion as they are much more efficient movement and are significantly less expensive than legged systems.

Domestic service personal robots are autonomous ground robotic vehicles equipped with specific purpose-implements that are meant to reduce the burden of a homeowner by performing tasks that also fall under the dull, dirty and dangerous criteria. These tasks may include lawn mowing, gardening, snow-removal, vacuuming, and carrying groceries. They could eventually be used for home security as well as home maintenance [3].

While the idea of personal robots may seem reasonable due to the presence of tasks that fall within the dull, dirty, dangerous criteria; they are often hard to justify economically. Typically, industrial robots are often quite expensive; however, they are also productive and their cost can be offset by the increase in quality they provide or the reduction in labor they may entail. On the other hand, personal service robots can be harder to economically justify if their cost is not offset by financial gain. As a result of this, most personal robots are often designed in a way that minimizes the cost to the consumer without sacrificing quality of the task being performed. However, this is often at the detriment to the robot's time efficiency.

Some of the popular service robots currently available in the market are shown in Figure 1.1.1 and Figure 1.1.2, i.e., the John Deere Tango E5 autonomous lawn-mower, [4], and the iRobot Roomba 960 autonomous vacuum cleaner [5]. Both of these robots utilize a ‘random walk’, visualized in Figure 1.1.3, to cover the task area. This is a very inefficient navigation method that does not guarantee to coverage of the intended area. However, the ‘random walk’ significantly reduces the cost of the robot as it removes the need for expensive localization sensors such as lasers, UHF trilateration, or GPS trilateration.

Figure 1.1: The John Deere Tango E5 can be seen in (1) and is an autonomous lawn-mower, [4]. The Roomba 960, as shown in (2), is an autonomous vacuum, [5]. Both of these robots utilize a ‘random walk’, shown in (3), which results in the inefficient completion of the task, [6], but at a large cost reduction to the cost of the robot.

While the Roomba 960 and the John Deere Tango E5 are both designed in a way that minimizes cost, they are still quite expensive with the Roomba costing close to $700.00 and the E5 at around $2,120. Also, these robots are designed to perform one task, as are most robots, [7]. This means that a homeowner who desires to automate tasks within the dull, dirty, and dangerous criteria will end up owning a fleet of unique robots which, in addition to being relatively expensive, all require maintenance, as well as storage space.

With this in mind there is the opportunity for a robotic vehicle to be designed that could fulfill more than one task having the potential to save cost, space, and increase the convenience to the user. This type of robotic system, which can perform more than one task, will be referred to here as a reconfigurable robot.

The areas of lawn-mowing and snow-removal are ideal tasks to be integrated into a reconfigurable system. As it stands, a lawn-mower/snow-plow hybrid has the potential to perform much more efficiently than the Tango E5 in Figure 1.1.1. The time it takes to complete the entire map of a ‘random-walk’ increase exponentially with the size of the map. For this reason the Tango E5 is currently sold in Europe where lawns are smaller than those in the US. By utilizing the guidance system of the snowplow the mower would be able to follow an efficient path, dramatically reducing the time of mowing while also reducing wasted energy.

Snow-plows often perform their work in low-friction environments, which means they need to have a steering-system that has a large amount of traction in order to effectively get enough torque to ground to push snow. With snow-plows, wheel slip during steering is not an issue. They must also be maneuverable enough to operate in the constrained space of an unplowed driveway. In contrast, robotic lawn-mowers require maneuverability in order to mow the intricate geometry patterns that are seen in many yards. While it does not matter for snow-plows, there must not be any wheel slip in robotic lawn-mowers as maneuvers would damage the lawn surface.

The focus of this research is on reconfigurable robotic vehicles for domestic personal services that include lawn-mowing, lawn-fertilization and snow-removal. Snow-removal requires large, heavy, robots in order to create the desired friction force to remove snow. In contrast, lawn-mowing and lawn-fertilization processes discourage the use of heavy robots to avoid damaging the lawn surface. While the snow-removal process can sweep the area is straight line segments, lawn operations are expected to follow irregular paths depending on the lawn design. In areas with large lawns, these robotic vehicles are expected to run at sufficiently high speeds so the job can be completed in the shortest possible time. Robots for both lawn operations and snow removal must be equipped with an accurate guidance system so that the task gets carried out in an organized manner for the job to be effective; additionally, they must be capable of adjusting their own speed by either accelerating or decelerating depending on the perception of the working environment.

These conditions require the robotic vehicle to not only be capable of accurately sensing the environment, with a powerful control algorithm, but also be highly maneuverable with wheels that offer sufficient traction on snow but do not damage grass. One element that guarantees maneuverability is the steering system. The next section provides an overview of existing steering systems and justification for the steering system used in this research.

1.2. Robot Steering Systems

This analysis will focus exclusively on wheeled mobile robots. Legged mobile robots have the potential to navigate through terrain that is inaccessible to wheeled vehicles, [8], which currently represents about half of earth's total landmass. However, they are also much less efficient than wheeled robots, [2], [7], and are also extremely difficult to implement and control, [7]. Figure 1.2.1 and Figure 1.2.2 show a few examples of legged mobile robots. Though these systems are extremely advanced, they are also impractically expensive for use as personal robots due to the large number of actuators, precision sensors, and powerful computers that are needed to perform simple tasks like maintaining static and dynamic balance.

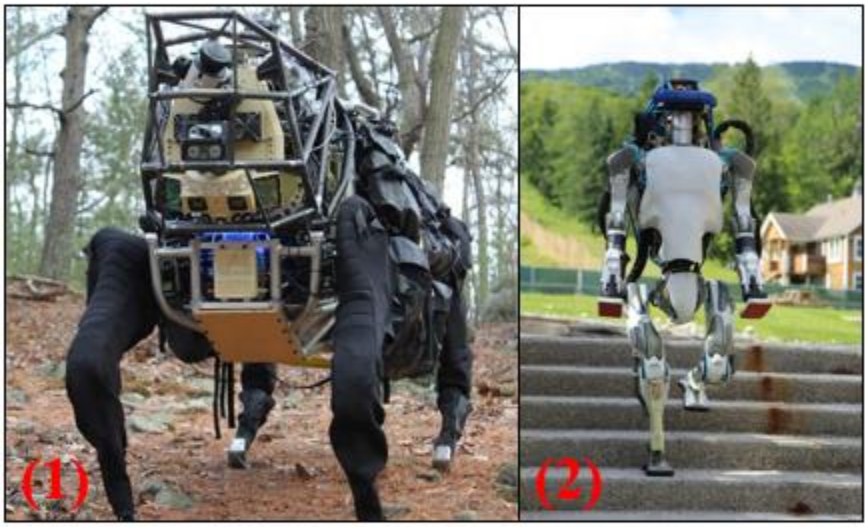

Figure 1.2: The LS3(Legged Squad Support System, [9, p. 3]) legged mobile robot by Boston Dynamics can be seen in (1), while Atlas, which can be seen in (2) and is also built by Boston Dynamics, represents their research in two-handed mobile manipulation, [10].

Steering systems for wheeled vehicles are often categorized by their performance in three different areas.

- Traction and Stability: The robot must have enough traction to maneuver through the indented terrain while also maintaining stability about the center of gravity [11].
- Maneuverability: The robot must be nimble enough to perform the desired task within the design space such that the quality of the task is not affected and the robot does not get stuck in the design space.
- Controllability: The robot must be controllable, in its motion, to follow the prescribed path to the desired degree of accuracy.

It is clear that there is no steering system that maximizes the qualities of maneuverability, controllability, and traction because the environments that robots face are unique to the task performed, [11], [12]. Also, high performance in a one area, often results in low performance in another. Some steering-systems that are highly controllable offer poor maneuverability while some steering-systems that are highly maneuverable have poor traction.

There are a large number of steering systems that have been developed for use on wheeled robotic vehicles. These include, differential-steering, skid-steering, two-wheel steer (2WS) Ackerman, four-wheel steer (4WS) Ackerman, articulating four-wheel drive, synchro-drive, Swedish-wheel omnidirectional, spherical-wheel omnidirectional, and four-wheel independent-drive/four-wheel independent-steer (4WD4WS).

1.2.1. Differential Steering System

Differential-steering is a steering-system that produces a motion vector by summing the individual wheel motions. This is a 2-DOF (degree-of-freedom) robot, as shown in Figure 1.3.1, in that the motion is a combination of longitudinal translation and yaw. Differential drive robots can be found in many industrial settings such as cleaning and sanitation, [13], as well as warehouse material transport, [14]. The Pioneer 3, shown in Figure 1.3.2, is a differentially steered robot that is used for many applications including: mapping, navigation, reconnaissance, vision, and manipulation, [15].

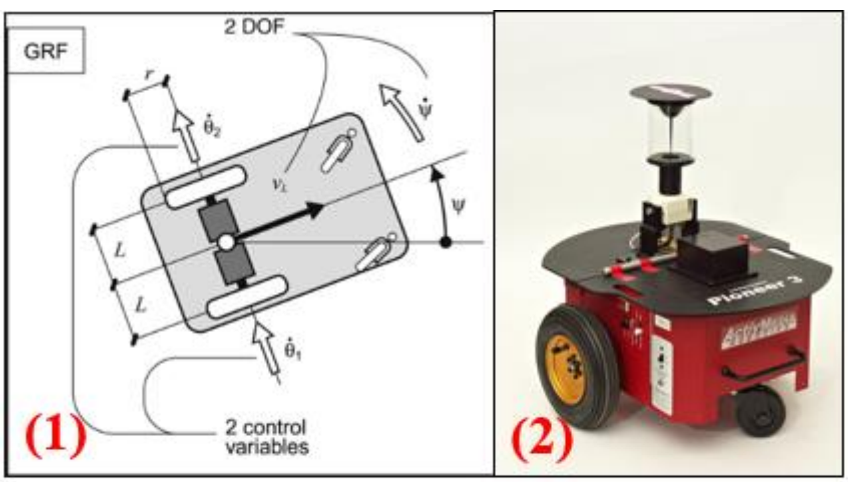

Figure 1.3: The kinematic model of a differential steering robot, [16],with two self-aligning castor wheels can be seen in (1) while the Pioneer 3, [17], differential steered robot can be seen in (2).

Differential-steering robots have a high degree of maneuverability, evident by their ability to do zero-turn maneuvers, [12]. If a circular frame is utilized the vehicle can maneuver in any space, that the frame can fit into, without getting stuck in that space. Also, the differential-steering system is highly controllable since it involves only two control signals and has no kinematic constraint to satisfy.

While differential-steering offers a high degree of maneuverability, it is vulnerable to performance errors that may be induced by lateral disturbances. For example, although lawn-mowing operations are likely to have less lateral disturbances, plowing snow at some blade angle is associated with a lateral reaction that differential steering cannot withstand. This is visualized in Figure 1.4. Additionally, this steering system works only on two wheeled robots with one or two casters. This limits the traction that can be achieved especially on slippery surfaces such as snow.

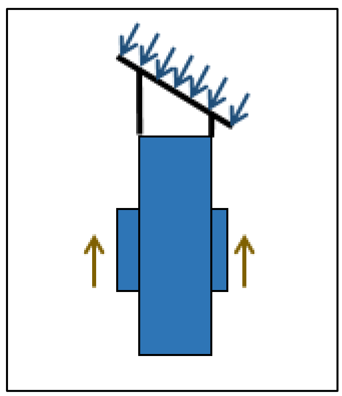

Figure 1.4: The differential steering robot cannot overcome lateral disturbances due to the longitudnal force distribution of the wheels

1.2.2. Skid Steering Systems

Skid-steering is similar to differential steering in that it performs curvilinear maneuvers by combining the velocity vectors of the wheels or tracks. Figure 1.5.1 shows how the velocity vectors of the robot define the yaw-rate of the robot. Skid steering robots, such as Seekur Jr in Figure 1.5.2, are often utilized for such applications as waste management, security, defense, and applications that involve harsh terrain navigation, [18].

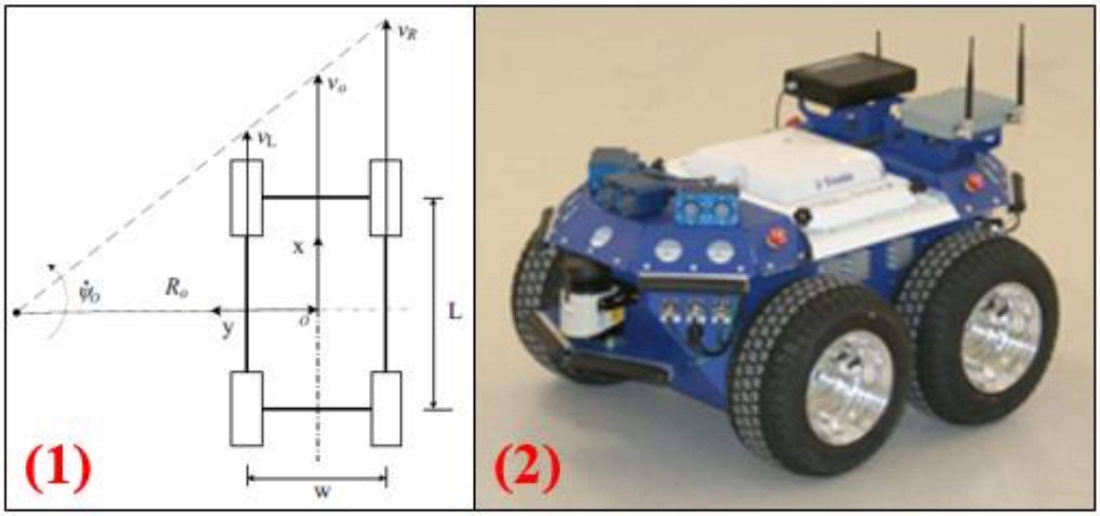

Figure 1.5: The kinematic model of a skid-steer robot, [19], is shown in (1) while the Seekur Jr, [20], skid steer robot can be seen in (2).

Skid-steering vehicles possess similar levels of maneuverability, especially on low friction surfaces, as differentially steered robots, and also offer high traction because of the increase in contact points, [12]. The maneuverability suffers as the surface friction increases due to the increase in torque required to ‘scratch’ the surface. In comparison to its counterpart, differential steering, this steering system offers some resistance to lateral disturbances if friction is sufficient. This is because of the counter-moment provided by the extra points of wheel-ground contact, shown in Figure 1.6.

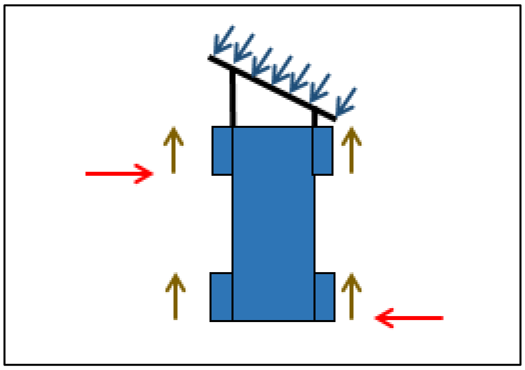

Figure 1.6: Skid-Steering robots are better able to handle lateral disturbances due to the counter moment created by the contact between the wheels and the ground.

While skid-steering has the advantage of being better suited for the high traction requirement of snow-removal, it is horribly unsuited for lawn-mowing. The reason for this is that every movement that a skid-steer robot makes involves slip. The combination of high traction and slip would tear up a lawn. Skid-steer robots are also difficult to model, kinematically and dynamically, because of the nonlinear relationships that exist between surfaces of uneven friction in conjunction with the vehicles curvilinear motion. Uneven friction between the left and right halves of the vehicle complicate navigation as the magnitude of the wheel slip is made up of induced slip from vehicle motion and unintended slip from uneven surface friction, [12]. Skid-steer vehicles are also inefficient, especially on surfaces of high friction, as the robot wastes energy while sliding the wheels against the ground while it is turning. This can have a high cost on the battery, [12], [21], and can even overcome the current limit of the motors or drivers [21].

1.2.3. Ackerman Two Wheel Steering Systems

Two-wheel steer (2WS) Ackerman is a steering system that utilizes two steered wheels in conjunction with two-wheel drive (2WD) or four-wheel drive (4WD). The wheels can either be mechanically linked or controlled by steer-by-wire in order to correctly negotiate turns. These steering systems are quite common in areas requiring long range transportation, [22]. The kinematic model for 2WS Ackerman steering, Figure 1.7.1, shows how the inner and outer steering tires do not have the same angle when steering. This angle offset is either accomplished with a tie-rod or steer-by-wire coordination. The GRP 2200 mobile robot which is used for many applications such as defense, material transport, mapping, and navigation, shown in Figure 1.7.2 uses this kind of steering, [23].

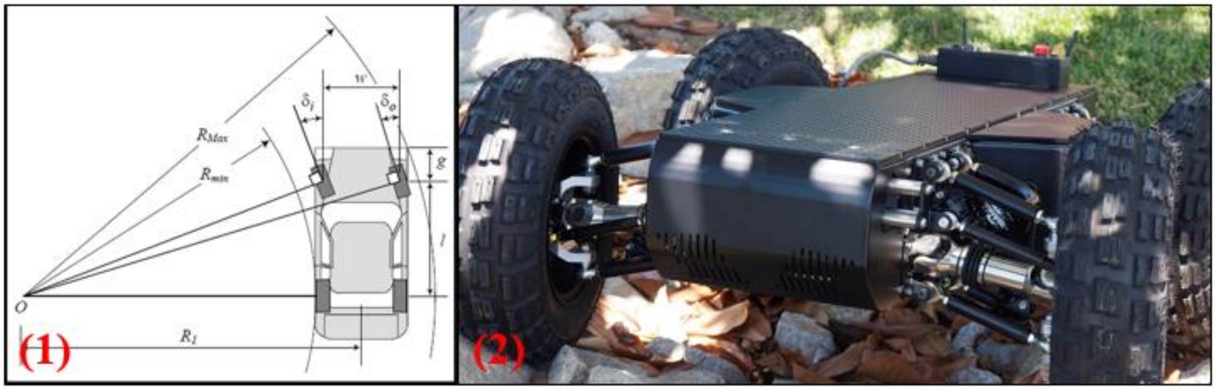

Figure 1.7: The kinematic model of a 2WS Ackerman robot, [22], can be seen in (1) while the GRP 2200 2WS mobile robot, created by Ambot, [21], can be seen in (2).

2WS Ackerman, which is an extremely common steering system in automobiles, has very good lateral stability during high speed turns, [11]. The system is also very controllable due to the simplicity of the kinematic and dynamic models, [19]. One reason for the dynamic simplicity, especially compared to skid-steer, is that there is little to no slip when maneuvering. Thus the wheel motion can be modeled using the pure-rolling condition.

While the 2WS Ackerman is highly controllable, it has extremely poor maneuverability at low speeds, due to the fact that the turning radius is larger than the vehicle itself, [11], [12], [24]. This is evident whenever lateral shifts and parking maneuvers are attempted as several changes in direction are often required, [11], [12]. During high speed maneuvers in automobiles the radius of curvature, on highways, is often large to avoid large centrifugal forces as well as the oversteer phenomenon, which means the limited steering radius is not a big issue. However, the reconfigurable robot would have severely limited lateral movement. Additionally, the drive wheel must be independently driven or utilize some form of differential in order to maintain the pure-rolling condition.

1.2.4. Ackerman Four Wheel Steering Systems

4WS Ackerman is different from 2WS Ackerman as there are steering inputs in both the front wheels and rear wheels. The system can be either 2WD or 4WD. The wheels can either be mechanically linked or controlled by steer-by-wire in order to maintain the ICR. Figure 1.8.1 shows the steering kinematics of 4WS Ackerman. Note how the ICR is perpendicular from the COG in Figure 1.8.1. Whenever Ackerman steering is used, the ICR is constrained to a perpendicular bisector from some point, often the midpoint between axles. Figure 1.9 shows how 4WS systems can also utilize 'crab' or 'in-phase' steering when a steer-by-wire system is used in order to translate without yaw. The Nomad robot, an experimental planetary rover shown in Figure 1.8.2, utilizes a steer-by-wire system that utilizes 4WS Ackerman kinematics.

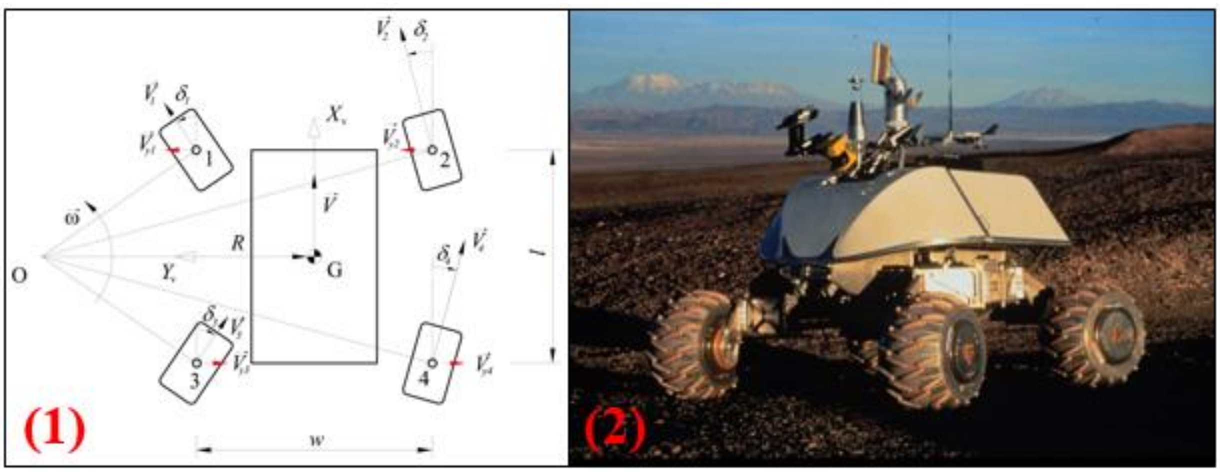

Figure 1.8: The kinematics of counter-phase steering, [25], can be seen in (1). The Nomad robot [25], which is designed to maneuver through planetary terrain, can be seen in (2).

4WS Ackerman has a significant advantage of maneuverability when compared to 2WS Ackerman due to the decreased radius of the ICR induced by the second steering angle, [26]. Additionally, when the front and rear steering are controlled distinctly the system has the advantage of utilizing in-phase steering which greatly increases high-speed straight line stability as the robot can maneuver laterally and longitudinally without yaw. This is desirable for countering the snow load while snow-plowing. The ability to utilize 4WD is also advantageous as the extra traction provided by two more drive motors is helpful for overcoming heavy snow loads.

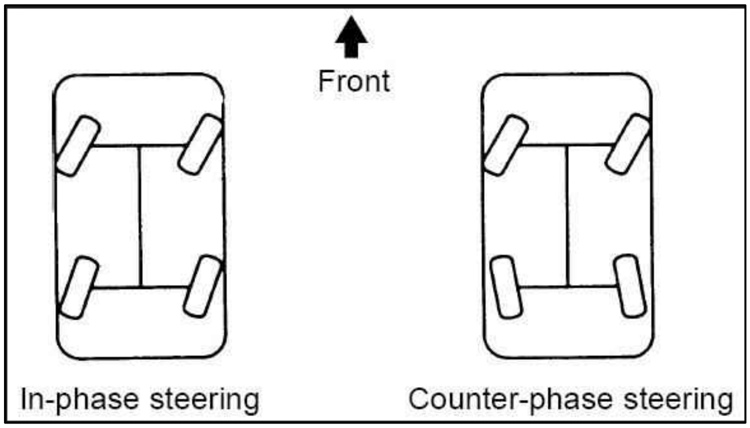

Figure 1.9: The difference between in-phase and counter-phase steering, [24].

4WS Ackerman does have some disadvantages. Though it can maneuver much better than its 2WS counterpart, it is still much less maneuverable than differential and skid steering. Furthermore, its ability to perform maneuvers that incorporate all 3-DOF should be taken with caution as it can only perform 2-DOF maneuvers at one time with combinations being translation motion (in-phase steering) or longitudinal and yaw motion (anti-phase steering). The system is also much more difficult to control than 2WS Ackerman, [26], as there can be as many as six inputs to control the multi-body system's nine DOF.

1.2.5. Articulating Steering Systems

Articulating robots are composed of two rigid bodies that actuate the yaw-angle between the two bodies, shown in Figure 1.10.1, to direct the motion of the robot, [27]. This is very similar to 4WS Ackerman in that the ICR is fixed on a perpendicular bisector of the robot. However, this bisector occurs at the joint of the two bodies as opposed to the midpoint of the front and rear axle of 4WS Ackerman robots. Articulating vehicles are often used in the agricultural, landscaping, forestry, and construction industries because of their high maneuverability, especially compared to the Ackerman system, [11], [27]. A robotic snow-plow concept utilizing articulating steering can be seen in Figure 1.10.2.

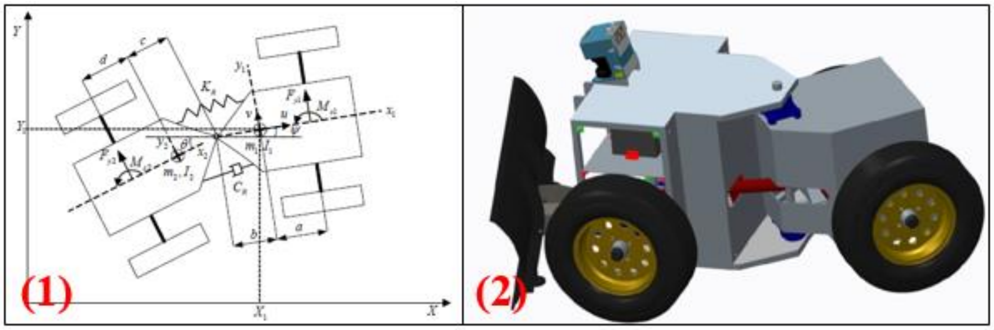

Figure 1.10: The kinematics of an articulating robot, [27], can be seen in (1) while the prototype of an articulating robotic snow-plow, [28], can be seen in (2).

Articulating vehicles often are 4WD and have large tires. This is advantageous in many rough-terrain industries because they often involve wet or slippery terrain. The big tires also have the added benefit of reducing the pitch motion of the vehicle, [27]. The articulating steering system often has wheel slip, regardless of differentials or independent wheel drive, due to the large width of the tires used. However, this is not a detriment as the industries in which it is often used have soil conditions that easily absorb the slip which prolongs tire life. Finally, the articulating vehicle has the added benefit of being more controllable than anti-phase 4WS Ackerman as it has one less control input since the front and rear bodies are mechanically linked.

The downfalls of the articulating steering system are that it is inherently unstable at high speeds. One of these instabilities occurs in the form of jackknifing, which happens when the front body tries to fold around the rear body. Additionally, the slip, which is not a detriment to the industries in which it is often employed, would wreck a lawn surface. Finally, lawn-mowing can often occur on steep road ditches. Articulating robots are very unstable, and could tip, when turning on steep slopes as the lateral wheel base gets skinnier when making sharp turns [11], [27].

1.2.6. Synchro-Drive Steering Systems

Synchro-drive is a steering system that maneuvers very similarly to anti-phase steering in that the motion vector is composed of a combination of lateral and longitudinal movement. Synchro-drive robots can be used as office robots, [29], as well as cleaning robots, [30]. Figure 1.11.1 shows how the steering and driving are coordinated with one motor each while a prototype synchro-drive mobile robot can be seen in Figure 1.11.2.

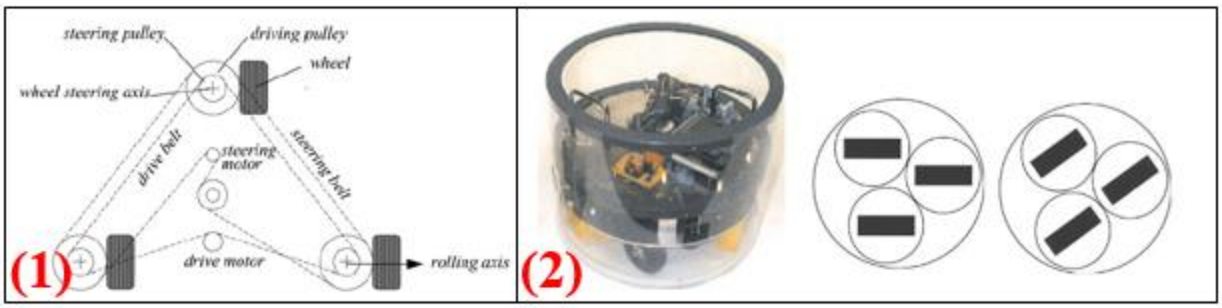

Figure 1.11: The mechanical linkages, [12], that dictate steering and driving of a synchro drive robot can be seen in (1) while a synchro-drive mobile robot, [30], can be seen in (2).

Synchro-drive is a highly controllable steering-system because there are only two inputs: wheel angle and wheel velocity. The drive wheels and steering angles are all coordinated by one motor each which utilizing chains and linkages. This makes straight-line motion of any kind a straight forward task as there is no need for a complex control structure in order to coordinate wheel velocities. The reason they are good at cleaning floors is due to the ‘complete coverage’ allowed by their purely translational movement [30].

While synchro-drive robots are easy to control, they are severely limited in their maneuverability in that they are not able to change heading which makes it an extremely poor candidate for the role of reconfigurable robot. Also, the mechanical linkage system that couples the steering and drive systems is very complex and requires very precise machining.

1.2.7. Omnidirectional Steering Systems

Swedish Wheels, shown in Figure 1.12.2 are a type of wheel that can be used to create an omnidirectional robot. Utilizing four Swedish Wheels, like the robot in Figure 1.12.3 with the wheel configuration of Figure 1.12.1, a robot can create motion from any combination of lateral, longitudinal, and yaw velocities. Robots with Swedish wheels are used for applications involving factory workshops, hospitals, elderly care facilities, and other areas with consistent, low-friction surfaces that utilize their high maneuverability in constrained environments, [31].

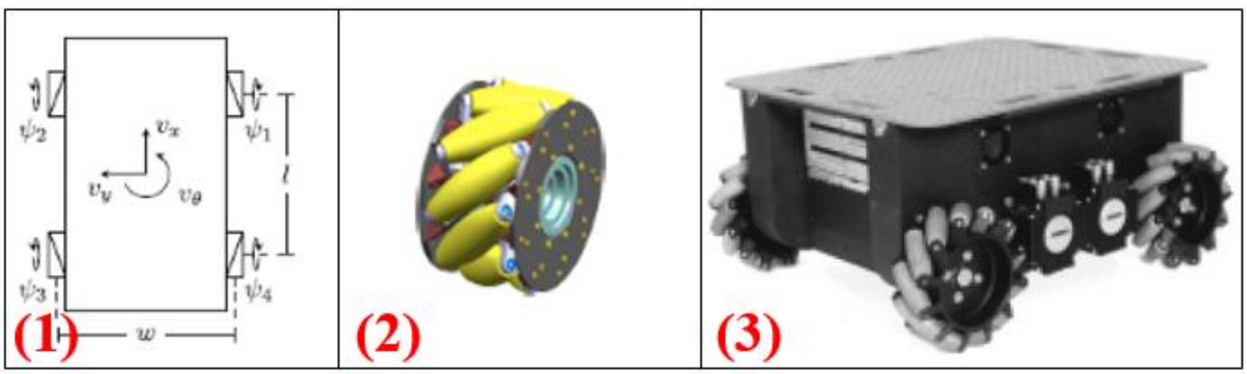

Figure 1.12: The kinematic model of a Swedish Wheel omnidirectional robot, [32], can be seen in (1). A 45° Swedish Wheel, [32], can be seen in (2). The Uranus omnidirectional robot, [11], with 45° Swedish Wheels can be seen in (3).

Omnidirectional robots are infinitely maneuverable on a plane surface. The system also only has four control inputs for the 3-DOF of the rigid body which means that it has less inputs and more control than other steering systems such as 4WS Ackerman [12]. Additionally, these robots do not have to reorient their wheels to turn which makes them truly omnidirectional when compared to other high maneuverability frames such as four-wheel independent-drive/four-wheel independent-steer, [33].

A robot utilizing Swedish wheels is undesirable for the position of reconfigurable robot because its motion occurs through sliding. Not only would this destroy grass, but omnidirectional wheels, such as Swedish Wheels, have very low friction with the surface which means that the steering-system would not be able to push snow. Though there are less control inputs than 4WS Ackerman, Swedish Wheel robots are much more difficult to control because of the nonholonomic nature inherent to sliding. Ground clearance is also limited because of a limited number of models available. Finally, they have trouble operating on uneven-surfaces where the castor is the only portion of the wheel in contact with the surface, therein negating the latitudinal vector created by the rollers [31].

Spherical Wheels produce motion vector by combining the motion of three wheels, positioned at the vertices of an equilateral triangle, with rollers producing motion at 120° from each other, as shown in Figure 1.13.1. The system is omnidirectional. A prototype robot, produced by EFPL, without the wheels can be seen in Figure 1.13.2.

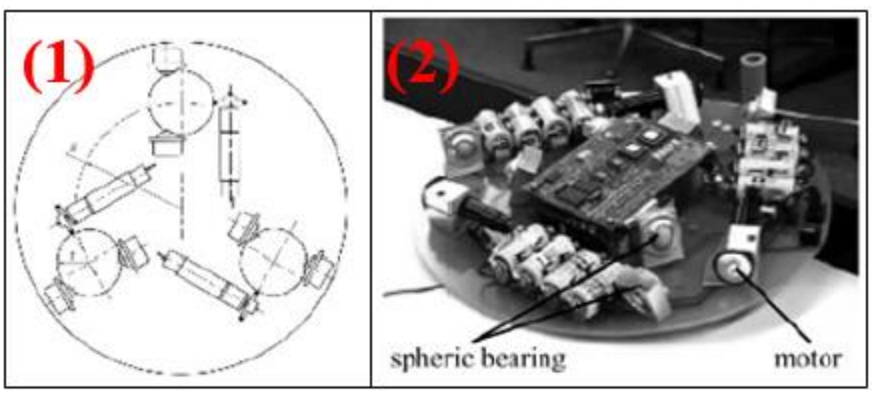

Figure 1.13: The roller/wheel layout as well as a prototype omnidirectional robot, named Tribolo, which uses spherical wheels can be seen in (1) and (2) respectively, [12].

Spherical Wheel robots have the same advantages that Swedish Wheel robots do, which is their infinite maneuverability. Spherical Wheeled systems are also slightly more controllable when compared to Swedish Wheel robots as their omnidirectionality is controlled with three inputs as opposed to four. The wheel design is also much less complex than Swedish Wheels.

However, like Swedish Wheel robots, their motion occurs through sliding which is not suitable to grass environments. In addition, they are also limited to small payloads and have extremely low ground clearance due to a limited number of spherical wheel sizes [12].

1.2.8. Four-Wheel Independent-Drive/Four-Wheel Independent-Steer

Four-Wheel Independent-Drive/Four-Wheel Independent-Steer (4WD4WS) system utilizes four drive motors to generate traction and four steering motors to control the heading direction of the wheels as illustrated in in Figure 1.14.1. This poses a strong challenge on meeting the kinematic constraints of driving around curved paths. The 4WD4WS steering-system has many advantages, especially the fact that it is most highly maneuverable steering system that can guarantee high tractions and reliability [34]. If the robot is viewed as a rigid body, then it is a highly over-actuated system utilizing eight control inputs to control 3-DOF. Their over-actuation and kinematic constraint characteristics have attracted strong research interest in recent days. A large number of robotic prototypes utilizing the 4WD4WS steering system have been developed due to the large amount of research that has been done on better controlling the structure. Robotic prototypes can be seen in Figure 1.14.2, Figure 1.14.3, Figure 1.15.1, Figure 1.15.2, and Figure 1.15.3.

Figure 1.14: The kinematic model, [35], depicting the steering angles with respect to the ICR can be seen in (1). The iMoro, ,[36] mobile platform can be seen in (2). BIBOT, by NDSU Mechanical Engineering, [37], can be seen in (3).

Figure 1.15: The GRP 4400 by Ambot can be seen in (1), Seekur, [38], by Omron Adept Mobile Robots can be seen in (2), and Hank by NDSU Mechanical Engineering can be seen in (3).

Additionally utilization of over actuated systems in conjunction with a flexible controller and high-speed response actuators, such as electric motors, means that the system has extremely good breaking and steering performance [39]. Vehicle stability can also be manipulated utilizing lateral dynamics as yaw moments can be counteracted by torque differentials between motors [39], [40]. The infinite maneuverability of omnidirectional-wheeled robots, the 4WD structure that improves the performance of plowing snow, in conjunction with no-slip make the 4WD4WS steering system the strongest candidate for the reconfigurable robot role.

However, the 4WD4WS system is also one of the most, if not the most, difficult systems to control. The reasons for this stem from the complex, nonlinear, multi-body dynamics that dictates the motion of the system as governed by the coordination of the many kinematic constraints which are required to maintain the no-slip condition.

1.3. Research Objectives

Owing to the nature of the tasks that need to be handled by the target robot, which involve variable loads, variable speeds and high maneuverability, it is recommended to employ a four-wheel independent drive, four-wheel independent-steer reconfigurable robotic vehicle. However, as it was noted early, this steering system is difficult to control because of its tight kinematic constraints, as illustrated in Figure 1.16, where if the path is constrained to go through the middle of the front and rear of the chassis, then the wheel speed and direction must satisfy the instantaneous center of rotation (ICR) condition. The most difficult part of control a 4WD4WS vehicle is on implementing the kinematic constraints in which the steering angle and wheel velocities are coordinated properly to satisfy the ICR conditions above, [36], [41]. Existing control algorithms circumvent this by measuring the yaw-rate and lateral velocity of the robot and assuming some constant longitudinal velocity, [42]–[44]. By using these measurements only, the individual wheel velocity vectors and the steering angles are determined. There can be considerable errors with this approach, [37] since the yaw-rate is an estimated state, the controller will always lag the vehicle motion. This means the system will always fail to satisfy the desired kinematic constraints, especially on curved paths.

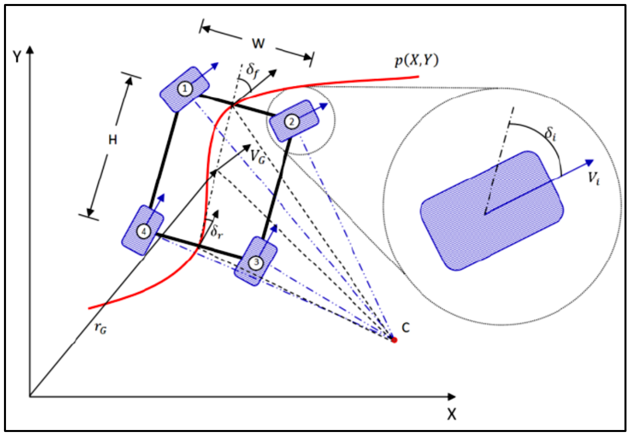

Figure 1.16: The steering angles of the robot constrained to the ICR by the path angles, [37].

A high fidelity dynamic model of the vehicle is required to develop an effective control algorithm for this system that can sustain variable loads and variable speeds while being effective in negotiating tight corners.

Although an extensive amount of research has been carried out to better understand and control this specific steering system: [36], [39], [41], [45]–[47], [47]–[69], most of it tend to make simplifying assumptions on the dynamics of the vehicle, which only apply to slow moving light weight robots and on paths with large curvatures. For example, some assume that angles between the robot and the path are very small, as seen in, [45], [49], [60], which demand large path curvatures. This assumption always has detrimental effects to the accuracy of path-tracking at large steering angles and tight curvatures [48]. There are some who make assumptions to allow some slip, especially when the terrain has extremely low friction, [60], [65]. The high traction required, and the large accelerations that may be involved make this an unreasonable assumption. Many models, such as: [36], [46], [69], [70], to name a few, are based on this assumption as it is valid when tractive forces have not been met, [57].

A common implementation, that allows some slip, sets the front pair of steering angles to be equal and also the rear pair of steering angles to be equal, [53]. As a result of this the front and rear steering angles can be described by one angle each in a model known as ‘bicycle’ or ‘single-track’. Again, this assumption is reasonable only in maneuvers involving a large radius of curvature, such as highways.

There are some robot controllers that avid the complex robot dynamics by relying on the robot kinematics only, [71]–[73]. By doing this the movement of the vehicle is equal to the actuator outputs. However, in order to maintain stability in the system the robot must be run at very low speeds while also avoiding large loads and high accelerations, [61], [65]. In this way Newton’s Second Law can be set to zero and the dynamics can be considered negligible. These conditions are not realistic for the target robot of this research especially since it involves large loads. High accelerations are also inevitable; the robot will be plowing large snow loads and undergoing large cornering forces in its operation.

These simplifications outlined above are normally done in order to maximize the characteristics of the 4WD4WS structure that are beneficial to the designed task whether that task be harvesting a field, [46], or controlling a vehicle on a highway, [45]. As discussed, these assumptions are inadequate since they don’t capture the actual robot dynamics. This research understands that the dynamics of any four wheel drive, four wheel steered vehicle can better be described using constrained Lagrangian formulation, which happens to be highly non-linear and non-holonomic. Therefore, the research has five objectives as follows.

1.3.1. Dynamic Modelling of a 4WD4WS Robotic Vehicle

There are many components of the 4WD4WS robot, however, for the purpose of dynamic modelling, the vehicle is assumed to be made of the main body (the robot’s chassis), and four wheel assemblies. The main body has momentum in the lateral, longitudinal, and yaw orientations while each of the wheel subsystems also has its own momentum in lateral, longitudinal, and yaw orientations similar to that of the chassis along with rotational momentum about the wheel axis and about the steering axis. The dynamics of the main body are dictated by the dynamics of each individual wheel system as the summation of the force vectors of each wheel subsystem results in the motion of the body. Each wheel subsystem needs to be coordinated in order to avoid wheel slip. This means that the force vectors on the body are not random and must be constrained by some fashion that enables the whole system to execute general plane motion stresslessly. On negotiation corners, a well-defined instantaneous center of rotation (ICR) must be satisfied by all parts of the robot.

1.3.2. Development of a Control Algorithm based on the Formulated Dynamic Model

The objective of any robot control algorithm to steer the robot to track the desired path. Many algorithms track the center of mass (CG) of the robot, and typically such algorithms always fail to track curved paths, since the if the wheels are to remain on the path, then CG will be off the path, and for CG to remain on the path, then the robot wheels must be off the path. This research intends to develop a control algorithm that tracks both the front and rear wheels while satisfying the high fidelity dynamic model of the robot. Both standard linear and nonlinear control methods will be evaluated and possibly fused to develop a new control algorithm suitable for the intended applications.

The environments of the reconfigurable system are unique. The robot must be traveling at a realistic speed and also change its speed to accommodate the curvature of the path in conjunction with the momentum of the robot. The robot must be able to make sharp turns. The robot must be able to maneuver when the tires have exceeded their saturation limit. These are conditions that both lawn-mowing and snow-plowing require regularly.

1.3.3. Numerical Validation of the Developed Control Algorithm

Before implementing the developed control algorithm on a real robot, it will be subjected to numerical validation steps through computer simulation. MATLAB environment will be used for this simulation.

1.3.4. Experimental Validation of the Developed Control Algorithm

The developed control algorithm will be experimentally tested on a real 4WS4WD robotic vehicle. This task has two parts, the first part is on developing a robotic vehicle equipped with the necessary sensors and actuators that fits the dynamics of the vehicle itself. The second part will be on coding the robot with the developed algorithm and run it on paths of various curvature at variable speeds and variable loads while tracking the wheels.

1.3.5. Thesis Write-up of the Results

This thesis is divided into five chapters. The next chapter will analyze the experimental vehicle that will be used to validate the model. Chapter three will discuss, in detail, the creation of the kinematic and dynamic models. Chapter four will discuss numerical simulations. Chapter five will discuss the experimental results. Chapter six will summarize the research.

2. THE EXPERIMENTAL 4WD4WS PROTOTYPE

2.1. Evolution of the Experimental 4WD4WS Prototype

The initial robotic vehicle, BIBOT-I, was used by Jonathan Nistler on his thesis research, [37]. However, Jonathon determined that the robot had several insufficiencies which limited the performance of the robot. He recommended that the IMU performance be improved as it was prone to dead-reckoning errors on the order of kilometers within a time span of one minute. He recommended that individual wheel encoders be added to provide feedback for wheel velocity. BIBOT-I utilized castor wheel on the center of the robot that provided absolute velocity of the COG with a resolution of 32 bits per revolution. Without individual wheel encoders wheel velocity had to be measured indirectly through the PWM output of the wheel controller, which is inaccurate. Another recommendation was that the control of the steering motors be improved such that the speed of the steering motor be variable as opposed to just on/off and position. The final recommendation was that GPS be added to improve the accuracy of the absolute position due to the fact that traveling farther than thirty meters resulted in large dead-reckoning errors.

With this in mind, BIBOT-I, Figure 2.1.1 and Figure 2.1.2, was stripped of its electronic control system and sensor system, as shown in Figure 2.5.1. Figure 2.2, Figure 2.3, and Figure 2.4 show the isometric view, the front and rear profile views, and the top and left profile views of the upgraded robot, Hank.

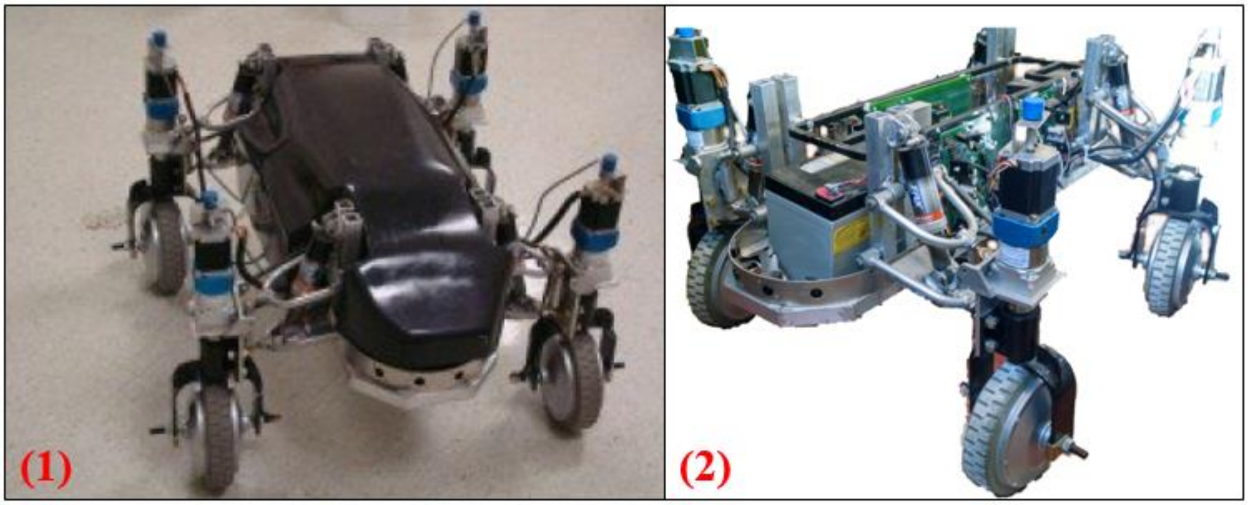

Figure 2.1: BIBOT-1, shown in (1), was the experimental robot used in Jonathan Nistler's thesis research, [37]. The robot’s batteries and printed circuit boards can be seen in (2).

Hank was designed to incorporate all of the recommended changes that Jonathan suggested in his research. A new IMU, with accuracy up to +/- 5° static yaw accuracy and +/-8° degree dynamic yaw accuracy has been added. Stepper motor drivers with direction, speed, and on/off control were implemented. Wheel encoders to provide velocity feedback were designed with resolution of 1024 bits to provide accurate velocity feedback. Finally, RTK GPS was implemented to provide position localization.

In addition to Jonathan’s recommendation, other systems were also updated. A SICK laser range finder replaced the original perception system which consisted of an array of sonar sensors. A 32 bit ARM CORETEX M4 STM32F407VG microcontroller, by STM electronics, as well as a RASBERRY PI 3B single board computer replaced the five dsPIC33FJMC128 PIC Microchip’s used in BIBOT. A fuse box was added that included a main fuse that incorporated all of the systems, as well as individual sub fuses, to ensure that no system was harmed by excessive current. Finally, powerful DC Motor Drivers were added that allowed for a variety of control and feedback options from the DC motors.

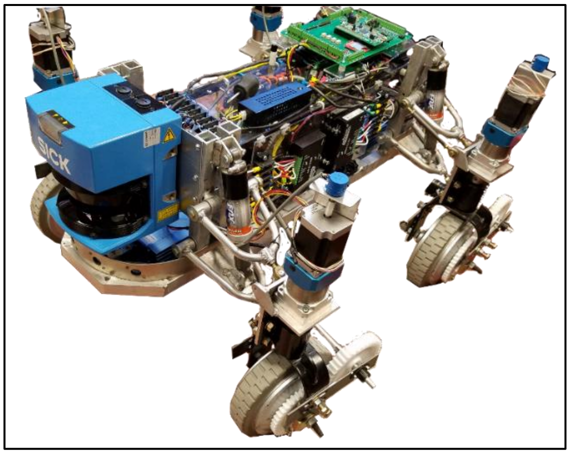

Figure 2.2: Hank is the upgraded version of BIBOT utilizing an advanced sensor and control system in order to increase the path tracking accuracy.

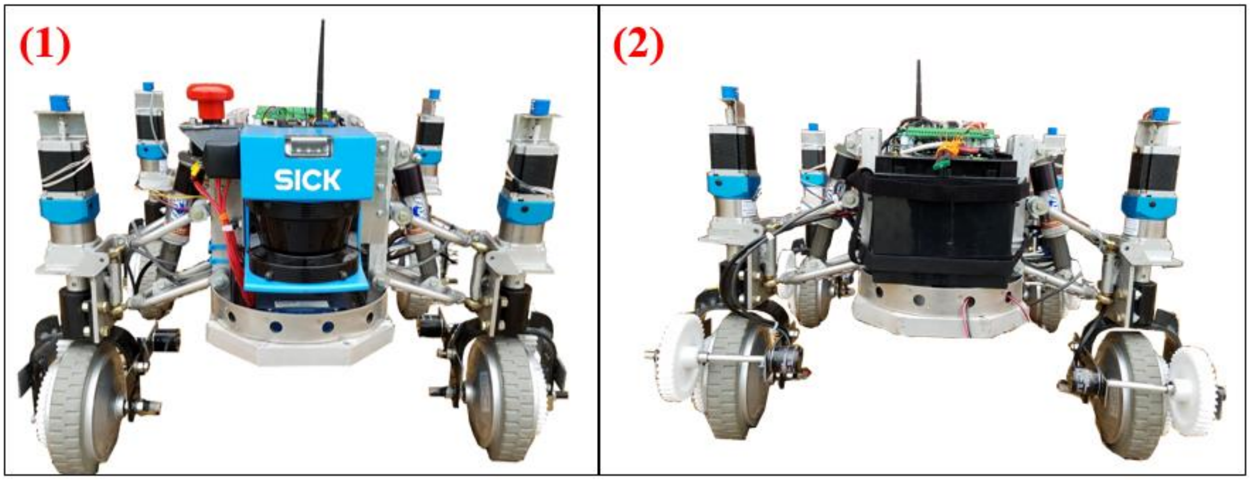

Figure 2.3: The front profile and rear profile views of the experimental system, Hank

Figure 2.4.1 and Figure 2.4.2 put emphasis on some of the new features that are utilized in the experimental system. Figure 2.4.1 shows the plastic gears, A, that are used in the encoder assembly. Additionally, Figure 2.4.1 also shows the stepper motors drivers, B. The drivers and encoder design are shown in further detail in Figure 2.5.2. Figure 2.4.2 labels several of the robots systems in including the laser perception system, C, the safety stop, D, the control system, E, the rotary encoders, F, and the buck-converter, G.

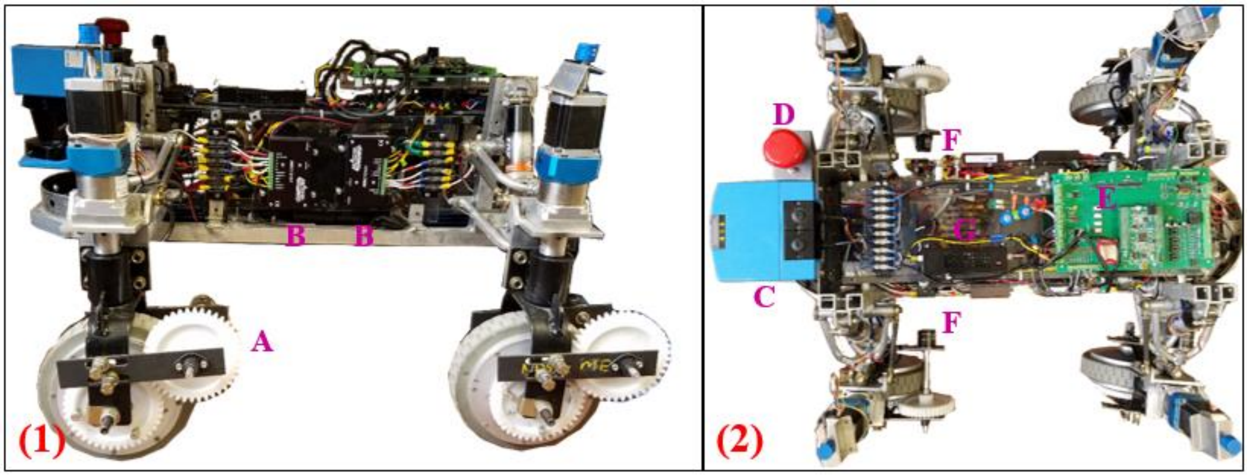

Figure 2.4: The left side, (1), and top, (2), views are shown of Hank, the experimental system. In (1), A, shows the plastic gears used to transfer wheel speed through a shaft to the encoders, F, which are shown in (2). Two of the four stepper motor drivers can be seen in B from (1). The sick laser, safety stop, control system, and buck-converter are shown in C, D, E, and G from (2) respectively.

2.2. Subsystems of the 4WD4WS Prototype

2.2.1. Main Chassis and Wheel Suspension

The chassis of the mobile robot, without sensors or motors, can be seen in Figure 2.5.1. The robots width and length are .75 meters and 1 meter respectively. There are four individual, yet identical, wheel steering/driving units on the vehicle, one of which is shown in Figure 2.5.2. Each steering/drive unit is independent of the others since the steering system is steer-by-wire. Each contact point needs a suspension system because of this. The suspension system is a ‘double wishbone’ which means two A-arms are used in tandem with a shock absorber.

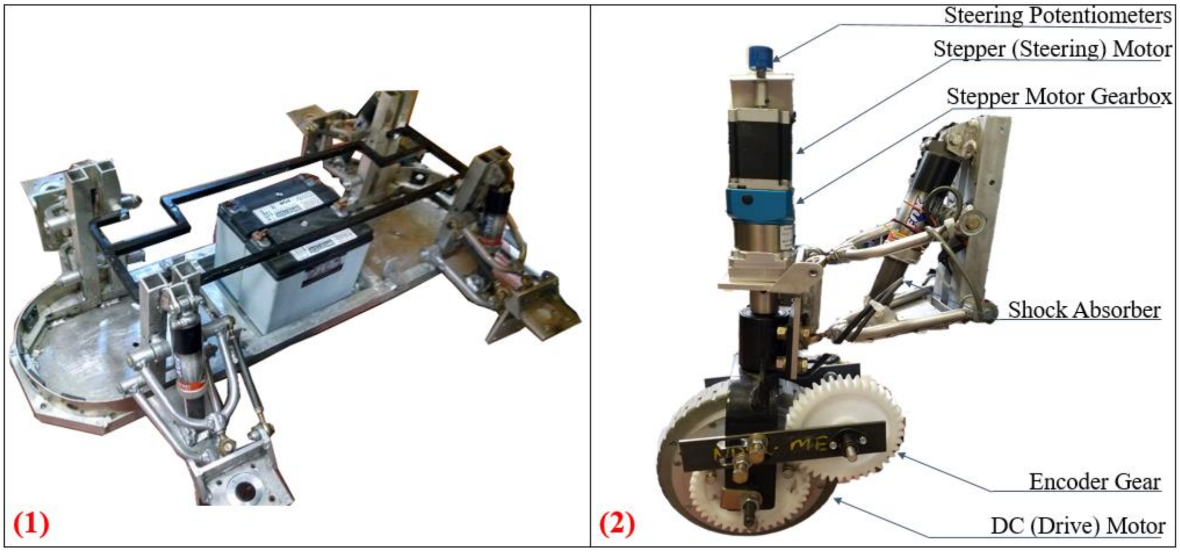

Figure 2.5: The chassis of the robot can be seen in (1). The steering, driving, and suspension system for each wheel can be seen in (2)

Two 12 Volt batteries were connected in series to provide power to the entire robot. A buck-converter was also installed to provide 12 Volt power while maintaining equal charge within the batteries. A fuse-box was constructed to protect the drivers, sensors, and control unit.

There are eight actuators on the robot comprised of four DC Motors and four stepper motors. The motors can be seen on the wheel units Figure 2.5.2. The DC Motors are connected to Roboteq drivers. While each of the DC Motors drivers can operate two channels, the stepper motor drivers can only control one motor each which means there are four of them total Figure 2.6 shows the graphical relationship between the power source, the drivers, and the motors. The drivers act as a buffer between the control system and the high voltage/current of the motors.

2.2.2. Power and Actuation

Two 12 Volt batteries were connected in series to provide power to the entire robot. A buck-converter was also installed to provide 12 Volt power while maintaining equal charge within the batteries. A fuse-box was constructed to protect the drivers, sensors, and control unit.

There are eight actuators on the robot comprised of four DC Motors and four stepper motors. The motors can be seen on the wheel units in Figure 2.5.2. The DC Motors are connected to Roboteq drivers. While each of the DC Motors drivers can operate two channels, the stepper motor drivers can only control one motor each which means there are four of them total. Figure 2.6 shows the graphical relationship between the power source, the drivers, and the motors. The drivers act as a buffer between the control system, which is a microcontroller, and the high voltage/current of the motors.

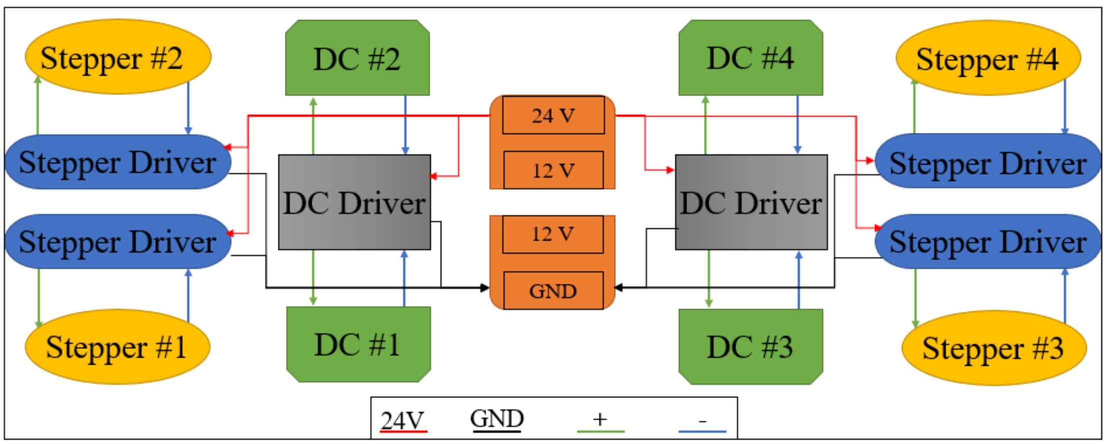

Figure 2.6: The actuators and their power source are shown. The Drivers receive a control signal from the microcontroller which dictates the output of the actuators.

2.2.3. Sensor Measurement

The experimental prototype has a significant sensor system in order to provide accurate feedback to the control system. Though the control model is constrained to six degrees-of-freedom, the robot's steer-by-wire system has a total of fifteen degrees-of-freedom. Eight of the fifteen DOF are actuators, and have their own feedback systems.

The potentiometers, seen in Figure 2.5.2 and Figure 2.7, are used to measure the steering angle of the stepper motors. Though feedback of stepper motors is uncommon, it has the advantage of not needing to 'home' the motors at startup. It also removes inaccuracies that can accumulate from counting steps if the motor slips. The potentiometers are a linear measurement, assuming the excitation voltage is constant, which means the relationship between angle and measured voltage can be found with a quick calibration. The control system uses an analog digital converter to create 16 bit resolution values for the potentiometer positions.

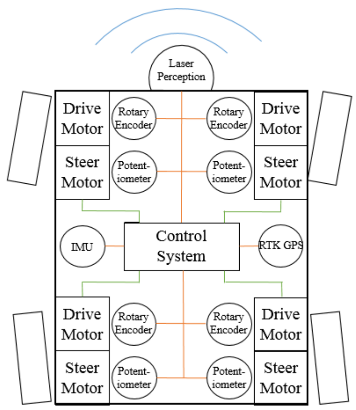

Figure 2.7: The Measurement and Actuation Systems of the robot are both connected to the Control System. The drive motors and steering motors, eight of the fifteen degrees-of-freedom, each have their own feedback sensors.

The DC Motors are hub motors which means the axles are rigidly locked to the forks of the suspension system. Most rotary encoders are attached to the axle of the measured apparatus, which is not possible for hub motors since the axle is fixed. An encoder measurement assembly was created in order to measure the wheel velocities of hub motors. A gear was attached to the hub motor which was connected to an idler gear. This idler gear connected to the encoder via a shaft. Bearings were used to minimize the torque exerted on the gears. The encoders are quadrature which means the control system can recognize positive or negative velocity. The encoder and gears can be seen in Figure 2.4.2. The resolution of the encoder is 1024 bits/revolution. It is straightforward to calculate velocity even though the encoder measure the change in position. This is done by dividing the traveled distance over a known time interval or vice versa. In either case, a constant relating the linear travel to the angular velocity, in bits per second, results in linear velocity.

A UM7 IMU was used to provide feedback for the yaw-rate of the robot. While the potentiometers and rotary encoders provided feedback for the individual wheel units, the IMU provides feedback for the robot's chassis. The IMU also has a magnetometer which means the yaw position can be verified as opposed to integrating the yaw acceleration twice, which is prone to significant round off error. The IMU uses SPI protocol to communicate with the control system.

A RTK GPS (Real Time Kinematics Global Positioning System) is used to provide absolute lateral and longitudinal position of the robot. Four satellites, as well as a reference position, are used in tandem to provide the user with latitude, longitude, and elevation information accurate to within a centimeter. The fourth satellite is used to provide a time reference to the system. The 'super user', as shown in Figure 2.8, increases the accuracy of the standalone GPS signal. It does this by computing the error of its known location from the GPS signal, it then relays that error to the 'user' who then corrects its position with the obtained error, [74].

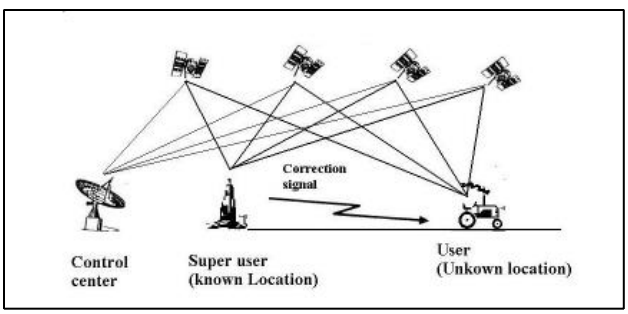

Figure 2.8: The relationship between the GPS satellites, the user, and the known location, [74].

While the IMU, potentiometers, and encoders that are used do provide very useful feedback, they are also prone to round off errors that accumulate with time due to integration. Figure 2.9.1 shows the estimated position of a robot following a path without GPS localization. It can be seen that the uncertainty of the robot's position grows as the robot progresses along the path as time increases. A similar path is shown in Figure 2.9.2, and has a much lower position uncertainty than in Figure 2.9.1, due to the fact that the system is able to provide absolute position checks that are not prone to the integration errors that accompany the other sensors.

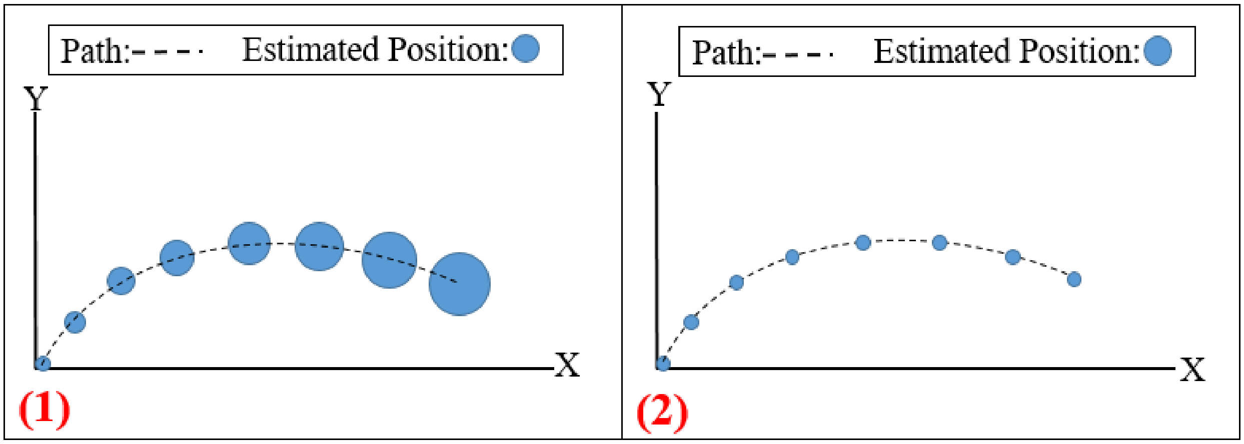

Figure 2.9: A robot following a path without global localization is shown in (1) while a robot following the same path with global localization is shown in (2).

2.2.4. Control System

The control system of the robot is made up of an STM32F4 CoretexM4 (STM) microcontroller and a RASBERRY PI3 (PI) single-board computer. The STM microcontroller was used to measure sensor data from the potentiometers, encoders, and IMU, as seen in Figure 2.10 and Figure 2.11

The microcontroller was used for prototyping this robotic system as they can offer a certain amount of flexibility, such as the large number of GPIO pins, timers, and wide range of communication protocols, which allow for changes during the development cycle. For example, the stepper drivers were frequency based which meant that standard PWM would not work as a changing input. To account for this, four timer channels were reconfigured to act as inputs for the stepper. All, in all, thirty-four pins from the microcontroller were utilized for the PCB in Figure 2.10. This includes eighteen pins for the sensor inputs and sixteen pins for the driver outputs. The single-core computer was used due to the ease in which it could deconstruct NMEA messages from the GPS, a task that would have required a large amount of programming from the microcontroller. Additionally, utilization of a computer negates the necessity of developing a real-time operating-system (RTOS) within the microcontroller. Due to the benefits and limitations of both systems, a hybrid control system was utilized to increase the programming and control efficiency.

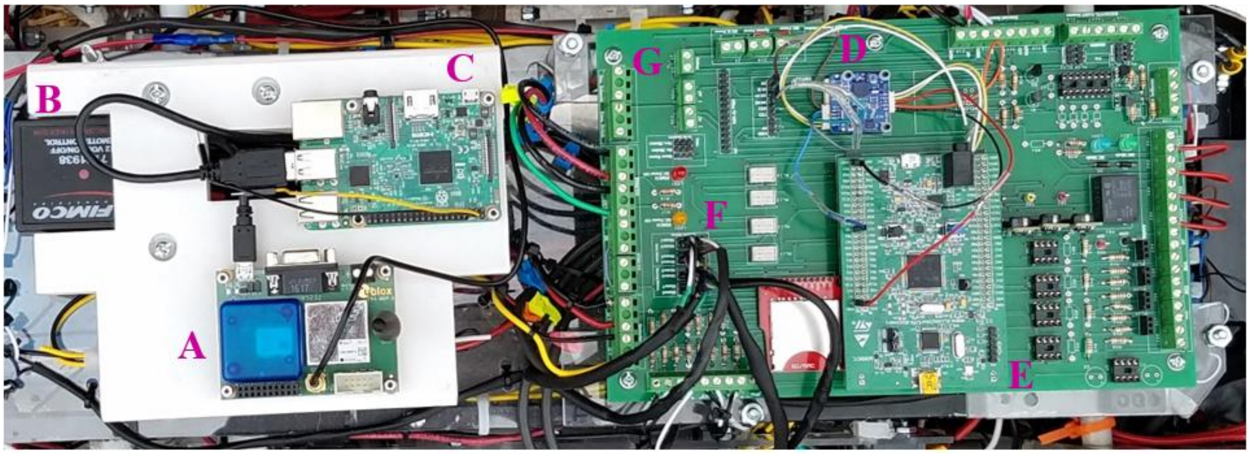

Figure 2.10: A shows the Rover RTK GPS, B shows the emergency safety stop, and C shows the RASBERRY PI 3B. D shows the IMU, E the STM32F4 microcontroller, and F shows the connection to the stepper and DC motors. G shows the PCB that was developed to connect the sensor and driver inputs to the microcontroller.

2.3. Architecture of the Robot Control System

Figure 2.11 is very helpful for understanding the relationship between the sensors, drivers, and processors. The STM measures wheel velocity via the encoders, steering angle via the potentiometers, and yaw-rate, as well as heading, from the IMU. The STM then sends this data to the PI using the MAVlink protocol. The PI, after decoding the RTK GPS data, combines the data from the sensors into an array, which represents the states of the robot. This array is logged to either a micro SD card or sent to a nearby computer over WIFI. The PI then uses the state data, as well as the dynamic equations of motion, in conjunction with a high-level controller to determine the necessary outputs for the systems actuators.

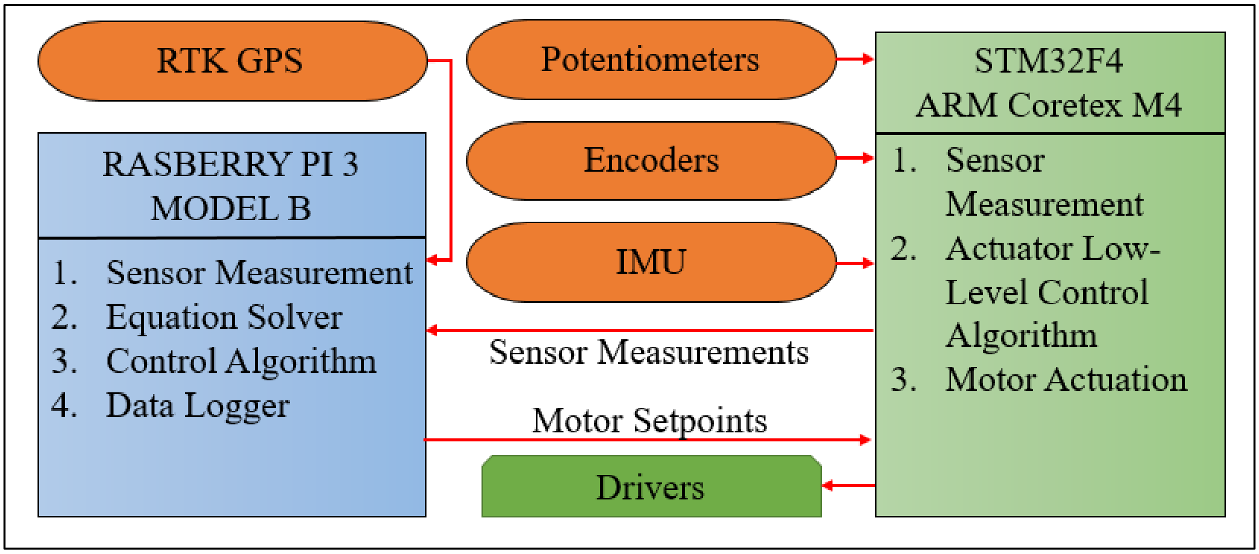

Figure 2.11: Relationship between the microcontrollers, sensors, and actuators

The PI, after determining the set-points for the steering and driving actuators, sends those values to each of the respective low-level controllers. The set-point for the stepper motors is steering angle while the setpoint for the DC motors is wheel velocity. This process continues until the target location has been reached.

3. KINEMATIC AND DYNAMIC MODELLING OF THE 4WD4WS ROBOTIC SYSTEM

3.1. Definition of Bodies and Coordinates

In developing this model, the first step considers the case of robotic motion in a 3D space following path coordinates defined using Cartesian Coordinates, P(X, Y, Z). The robot body is assumed to be homogenous with its Center-of-Gravity (COG), located in the center of the robot with respect to the height and width, as shown in Figure 3.1.

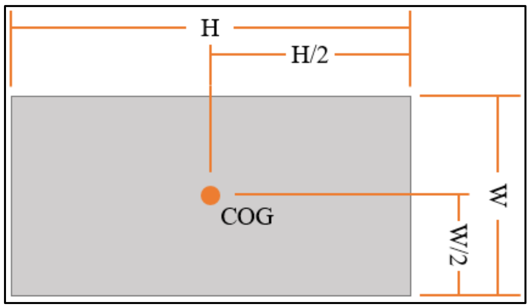

Figure 3.1: The location of the COG with respect to the dimensions of the robot.

A four wheeled vehicle is a multi-body system composed of five bodies, i.e., four wheel bodies as well as the chassis. The position and orientation of each body can be defined using each body's inertial coordinates and Euler Angles. Figure 3.2 shows the five bodies and the coordinates that define each body. Under this assumption, the robot vehicle system seems to have thirty degrees of freedom.

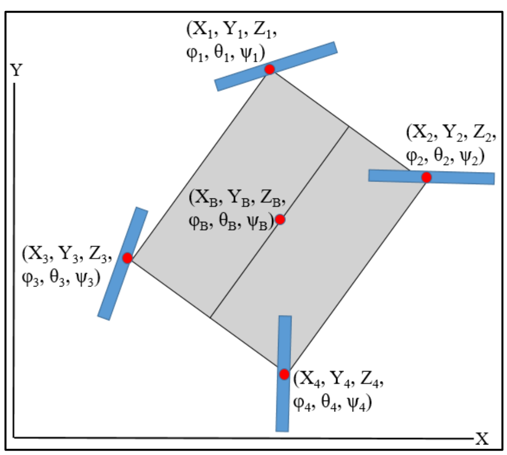

Figure 3.2: The coordinates that define the 6 Degree-of-Freedom pose of each body.

However, normal surface ground vehicles are constrained to flat horizontal plane motion which means the vertical, Z, pitch, φ, and roll, θ, motion of each body don't come into play. This reduces the model's 30 DOF to 15 DOF. As shown in Figure 3.3, are all defined in the global frame.

The primary problem in controlling these vehicles lies in the required coordination of the wheel drive speed, and steering angle to avoid, not only, wheel-slip, but also structural damage due to overstressing the axles. This means that the orientation of each wheel body is not arbitrary. While there may be 15 coordinates necessary to define the pose of each body, the required coordination of the wheel bodies means the system motion must have less than 15 DOF to allow for the constraint above to be satisfied.

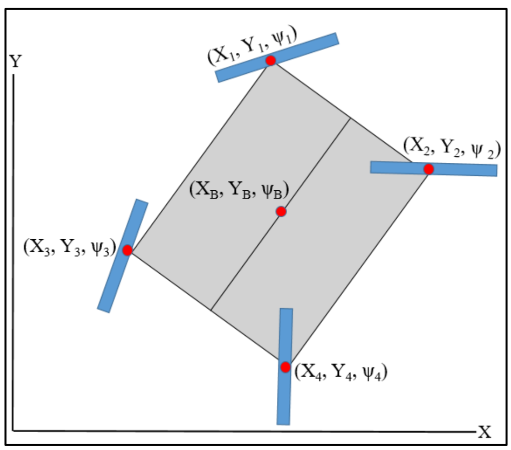

Figure 3.3: The coordinates that define the 3 Degree-of-Freedom pose of each body.

3.2. Path Tracking: The Relationship between the Path and Chassis Position

The common method of path-tracking for 4WD4WS robots is to track the path through the chassis' center-of-gravity while simultaneously controlling the orientation of the robot. This is so that positive obstacles, obstacles that protrude from the ground, are avoided, [37]. This tactic is common in research due to the simplicity offered by defining the motion relative to the robot's COG, which, in turn, increases the flexibility in the robot being able to determine its orientation and position. This helps in achieving obstacle avoidance capabilities. However, this method of path-tracking, shown in Figure 3.4, includes the possibility of the wheel bodies driving off the road, even while the COG stays on the path.

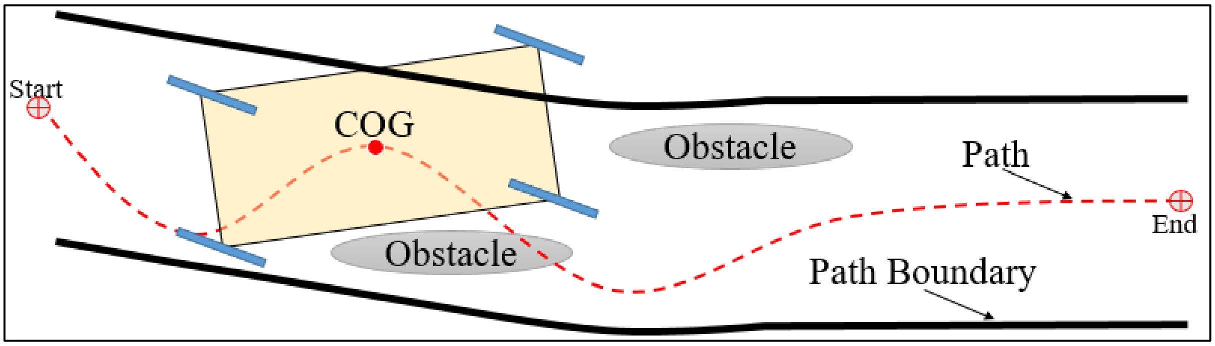

Figure 3.4: Path-Tracking through the COG of the robot. The orientation is flexibile such that positive obstacles are avoided. Driving off the path with a wheel body is a possible consequence of COG tracking.

This research employs another method of path-tracking where the midpoint of the front and rear of the chassis have to remain on the center of the path, as shown in Figure 3.5. This method of path-tracking was proposed earlier in, [37], however, at the time of its introduction, the model that was developed was based off geometric constraints only. Under this approach, the robot will never drive outside the path-boundary if the path is predetermined with that goal in mind.

Figure 3.6 shows that the pose of the chassis is determined at the COG. However, Figure 3.5 shows that the COG is independent of the path when tracking through the midpoint of the front and rear. This is problematic as the position of the robot relative to the path should always be known in order to track it effectively.

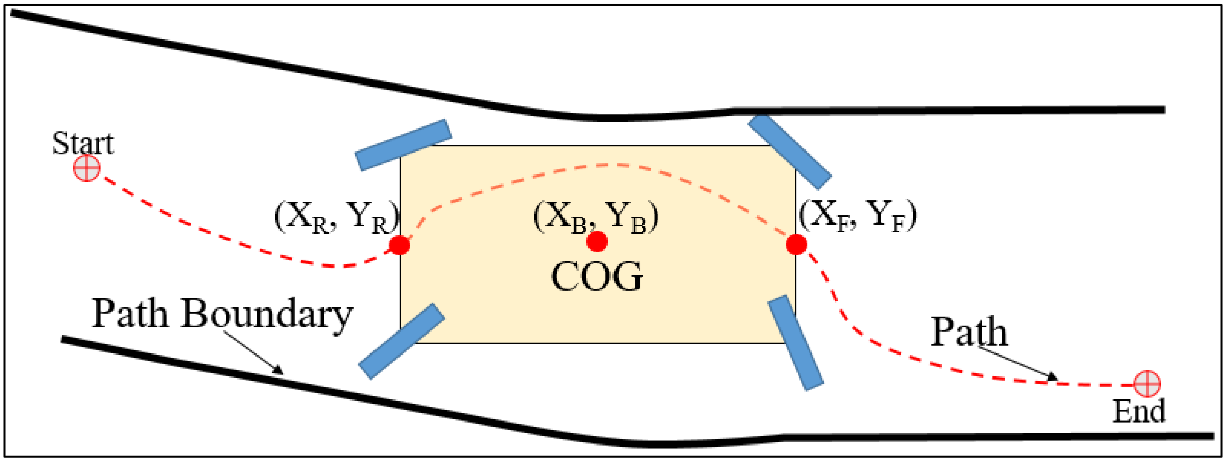

Figure 3.5: Path-Tracking through the midpoints of the front and rear of the robot. The path should be determined from the path boundary such that the robot never drives into negative obstacles.

Since the dynamics of the vehicle require its COG to be known, under the proposed path tracking approach, the location of the COG will be defined by the midpoints of the front and rear of the robot such that,

$$X_B = \frac{X_F + X_R}{2}, \tag{3.1}$$

$$Y_B = \frac{Y_F + Y_R}{2}, \tag{3.2}$$

$$\tan(\Psi_B) = \frac{Y_F - Y_R}{X_F - X_R}, \tag{3.3}$$

where X_B, Y_B, and Ψ_B define the pose of the chassis, as illustrated in Figure 3.6, and X_F, Y_F, X_R, and Y_R are the Cartesian Coordinates that define the position of the front and rear midpoints of the chassis will be referred to, hereafter, as 'path coordinates'.

The position of the four wheel bodies, also shown in Figure 3.6, can easily be derived in terms of the path coordinates if the pose of the chassis is known. The wheel bodies are permanently connected to the frame which means their position can be related to the position of the COG.

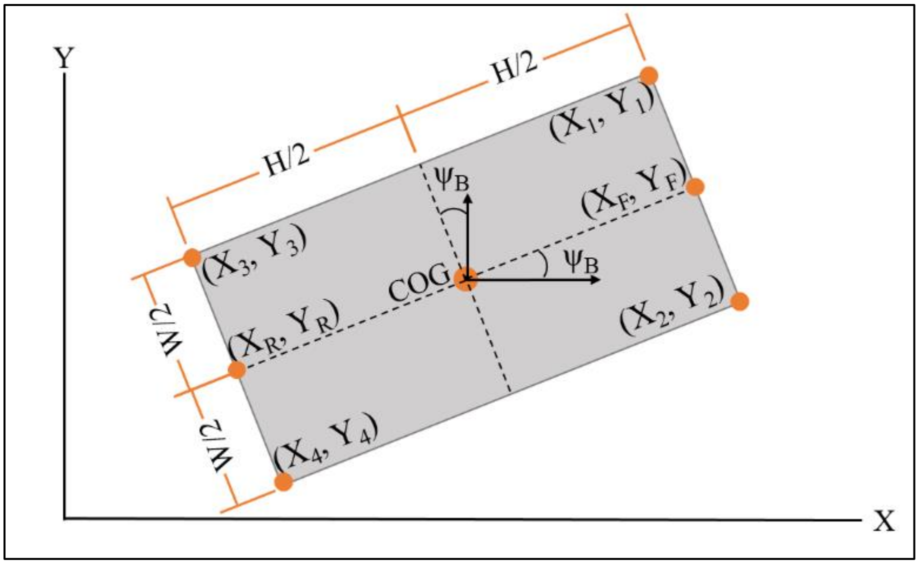

Figure 3.6: The relationship between the pose of the chassis, the path coordinates, and the wheel positions.

In particular, the position of each wheel body, for the pose shown in Figure 3.6, then becomes,

$$X_1 = X_B + \frac{H}{2}\cos(\psi_B) - \frac{W}{2}\sin(\psi_B), \tag{3.4}$$

$$Y_1 = Y_B + \frac{H}{2}\cos(\psi_B) + \frac{W}{2}\sin(\psi_B), \tag{3.5}$$

$$X_2 = X_B + \frac{H}{2}\cos(\psi_B) + \frac{W}{2}\sin(\psi_B), \tag{3.6}$$

$$Y_2 = Y_B + \frac{H}{2}\cos(\psi_B) - \frac{W}{2}\sin(\psi_B), \tag{3.7}$$

$$X_3 = X_B - \frac{H}{2}\cos(\psi_B) - \frac{W}{2}\sin(\psi_B), \tag{3.8}$$

$$Y_3 = Y_B - \frac{H}{2}\cos(\psi_B) + \frac{W}{2}\sin(\psi_B), \tag{3.9}$$

$$X_4 = X_B - \frac{H}{2}\cos(\psi_B) + \frac{W}{2}\sin(\psi_B), \tag{3.10}$$

$$Y_4 = Y_B - \frac{H}{2}\cos(\psi_B) - \frac{W}{2}\sin(\psi_B). \tag{3.11}$$

Equations (3.4)-(3.11) show the individual position components of each wheel body with respect to the pose of the chassis. These equations can be further modified such that they are with respect to the path coordinates. First, the relationship between sine, cosine, and tangent with respect to the path coordinates must be known, where

$$\tan(\psi_B) = \frac{\sin(\psi_B)}{\cos(\psi_B)}, \tag{3.12}$$

$$\cos(\psi_B) = \frac{X_F - X_R}{H}, \tag{3.13}$$

$$\sin(\psi_B) = \frac{Y_F - Y_R}{H}. \tag{3.14}$$

Using the relationships in (3.13) and (3.14), Equations (3.4)-(3.11) can be replaced with,

$$X_1 = \frac{X_F + X_R}{2} + \frac{H}{2H}(X_F - X_R) - \frac{W}{2H}(Y_F - Y_R), \tag{3.15}$$

$$Y_1 = \frac{Y_F + Y_R}{2} + \frac{H}{2H}(X_F - X_R) + \frac{W}{2H}(Y_F - Y_R), \tag{3.16}$$

$$X_2 = \frac{X_F + X_R}{2} + \frac{H}{2H}(X_F - X_R) + \frac{W}{2H}(Y_F - Y_R), \tag{3.17}$$

$$Y_2 = \frac{Y_F + Y_R}{2} + \frac{H}{2H}(X_F - X_R) - \frac{W}{2H}(Y_F - Y_R), \tag{3.18}$$

$$X_3 = \frac{X_F + X_R}{2} - \frac{H}{2H}(X_F - X_R) - \frac{W}{2H}(Y_F - Y_R), \tag{3.19}$$

$$Y_3 = \frac{Y_F + Y_R}{2} - \frac{H}{2H}(X_F - X_R) + \frac{W}{2H}(Y_F - Y_R), \tag{3.20}$$

$$X_4 = \frac{X_F + X_R}{2} - \frac{H}{2H}(X_F - X_R) + \frac{W}{2H}(Y_F - Y_R), \tag{3.21}$$

$$Y_4 = \frac{Y_F + Y_R}{2} - \frac{H}{2H}(X_F - X_R) - \frac{W}{2H}(Y_F - Y_R). \tag{3.22}$$

These equations define the orientation of the body, and positions of all bodies in the system using the path coordinates only. The next task is to define the wheel orientations using these path coordinates. To do so, the instantaneous center of rotation, ICR, must be known.

3.3. Relationship between the Path and the ICR

The steering angles of the wheels and the drive wheel velocities need to be coordinated based off the ICR such that wheel slip is avoided. However, the position of the ICR is infinitely variable on a 2D plane which means that its position must first be determined before the wheel's orientations and velocities can be determined. Figure 3.7 shows that the angle between the perpendicular bisector of the width and a line tangent to the path can be formed at both the front and rear of the robot. These angles are labeled δ_F and δ_R. Line segments, labeled ρ_F and ρ_R, extend perpendicular from the path angles, δ_F and δ_R. The intersection of the radius', ρ_F and ρ_R, is the location of the ICR.

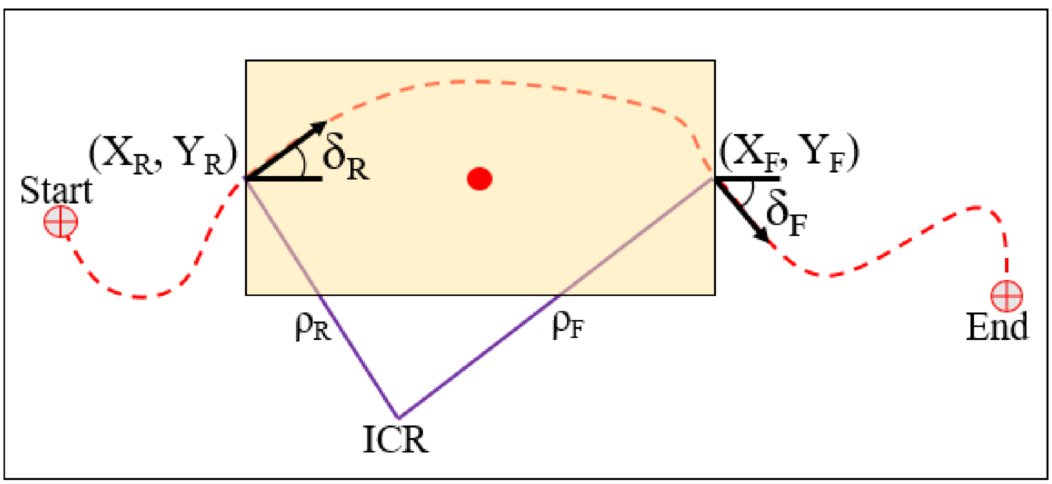

Figure 3.7: The relationship between the path and the position of the ICR.

The ICR position can be found using a method called Intersection of Two Circles, [75]. Figure 3.8 shows the robot at an arbitrary angle, ψ_B, with an ICR located at (X_0, Y_0). By imagining a triangle formed with the height, H, and the path ICR radius's, ρ_F and ρ_R, the third point, (X_0, Y_0), can be calculated by the known points (X_F, Y_F) and (X_R, Y_R).

The values, F and V, as shown in Figure 3.8, are,

$$F = \frac{\rho_F^2 - \rho_R^2 + H^2}{2H}, \tag{3.23}$$

$$V = \rho_F \cos(\delta_F). \tag{3.24}$$

With these values, X_0 and Y_0 become,

$$X_0 = X_F - F\cos(\psi_B) + V\sin(\psi_B), \tag{3.25}$$

$$Y_0 = Y_F - F\sin(\psi_B) - V\cos(\psi_B). \tag{3.26}$$

These equations can be simplified to using, (3.12), (3.23), (3.24), (3.41), (3.42) to create,

$$X_0 = X_F - \cos(\delta_R)\sin(\delta_F)\csc(\delta_R + \delta_F)\,(X_F - X_R) + \frac{\cos(\delta_R)\cos(\delta_F)}{\sin(\delta_F+\delta_R)}(Y_F - Y_R), \tag{3.27}$$

$$Y_0 = Y_F - \cos(\delta_R)\sin(\delta_F)\csc(\delta_R + \delta_F)\,(Y_F - Y_R) - \frac{\cos(\delta_R)\cos(\delta_F)}{\sin(\delta_F+\delta_R)}(X_F - X_R). \tag{3.28}$$

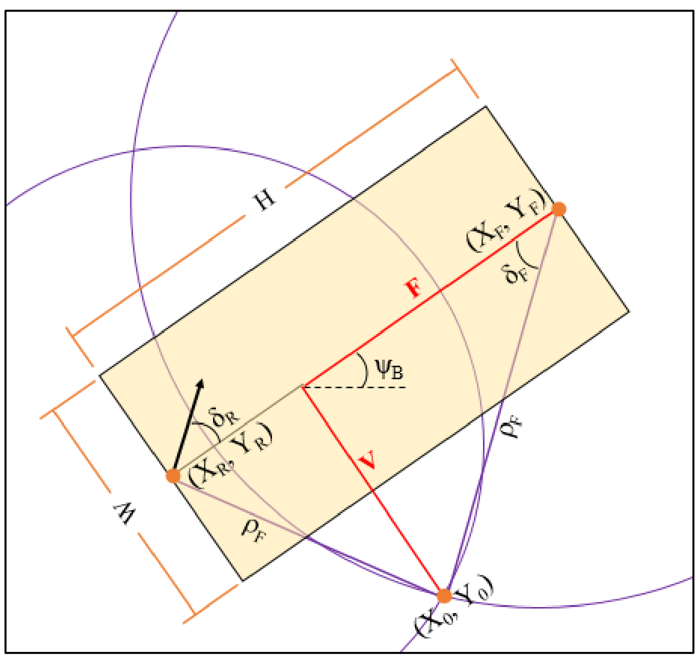

Figure 3.8: Finding the vertex position of a triangle using known points. The method is called "Intersection of Two Circles", [75].

3.4. Relationship between the ICR and the Steering Angles

With the location of the ICR identified, the steering angles of each wheel body can be found using trigonometric laws such as the Law of Sines and the Law of Cosines. To make use of these laws, the relative wheel angles, δ_1, δ_2, δ_3, δ_4, are defined to relate the wheel angle, Ψ_i, and the body angle, Ψ_B as

$$\Psi_1 = \Psi_B + \delta_1, \tag{3.29}$$

$$\Psi_2 = \Psi_B + \delta_2, \tag{3.30}$$

$$\Psi_3 = \Psi_B + \delta_3, \tag{3.31}$$

$$\Psi_4 = \Psi_B + \delta_4, \tag{3.32}$$

as shown in Figure 3.9, where the yaw-orientations, Ψ_i, are measured counterclockwise from the global, X, axis. However, the steering angle orientations are defined clockwise for angles, δ_1 and δ_2, illustrated in Equations (3.29) and (3.30), and counterclockwise for angles, δ_3 and δ_4, illustrated in Equations (3.31) and (3.32).

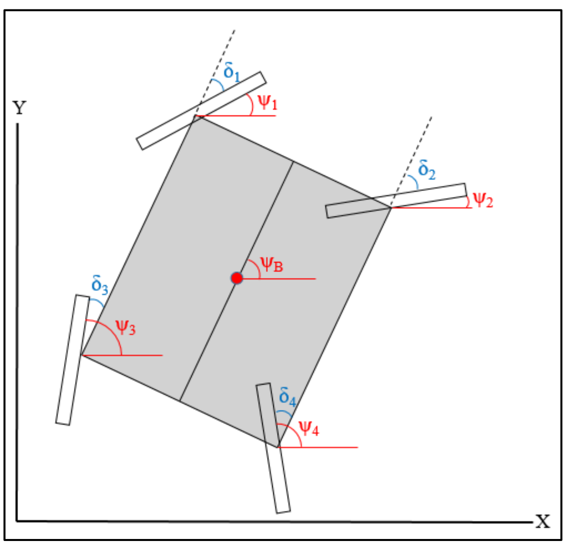

Figure 3.9: The relationship between the global orientation and steering angle of the wheel bodies.

Figure 3.10 and Figure 3.11 reveals how these triangles can be constructed to include the ICR and the sides of the robot, W and H. Also included are the steering angles, δ_1 and δ_2, the path angles, δ_F and δ_R, as well as the ICR radius, ρ_1, ρ_2, ρ_F and ρ_R. Figure 3.10 deals exclusively with the front steering wheels while Figure 3.11 can be used to determine the angle of the rear steering wheels,

$$\rho_1 \sin(\delta_1) = \rho_F \sin(\delta_F), \tag{3.33}$$

$$\rho_2 \sin(\delta_2) = \rho_F \sin(\delta_F), \tag{3.34}$$

$$\rho_3 \sin(\delta_3) = \rho_R \sin(\delta_R), \tag{3.35}$$

$$\rho_4 \sin(\delta_4) = \rho_R \sin(\delta_R). \tag{3.36}$$

Equations (3.33)-(3.36) show the initial form of the law-of-sines. These equations will be used later when the time derivative of the wheel steering angles are required for Lagrangian Dynamics.

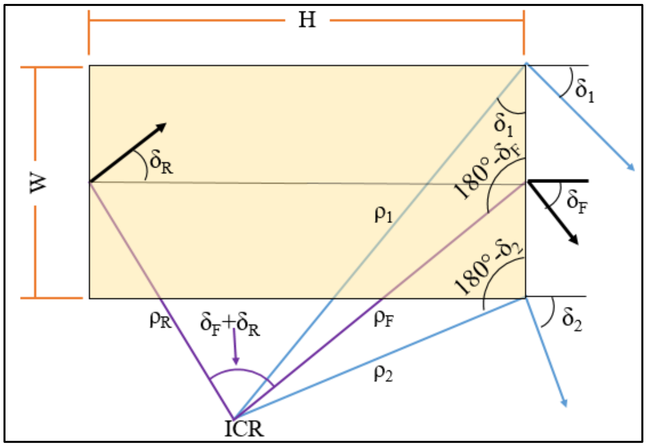

Figure 3.10: Triangular geometry used to determine steering angles 1 and 2.

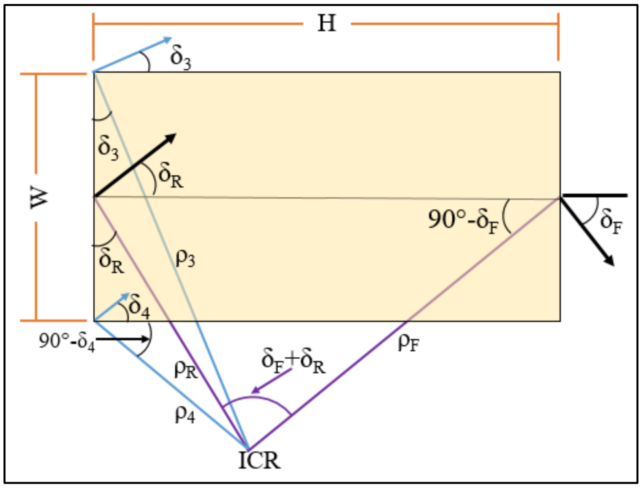

Figure 3.11: Triangular geometry used to determine steering angles 3 and 4.

The front and rear pairs of steering angles can be found by rearranging Equations (3.33)-(3.36),

$$\delta_1 = \sin^{-1}\left(\frac{\rho_F \sin(\delta_F)}{\rho_1}\right), \tag{3.37}$$

$$\delta_2 = \sin^{-1}\left(\frac{\rho_F \sin(\delta_F)}{\rho_2}\right), \tag{3.38}$$

$$\delta_3 = \sin^{-1}\left(\frac{\rho_R \sin(\delta_R)}{\rho_3}\right), \tag{3.39}$$

$$\delta_4 = \sin^{-1}\left(\frac{\rho_R \sin(\delta_R)}{\rho_4}\right). \tag{3.40}$$

Equations (3.37) and (3.38), corresponding to Figure 3.10, represent the front pair of steering angles, δ_1 and δ_2, whereas Equations (3.39) and (3.40), corresponding to Figure 3.11, represent the rear pair of steering angles, δ_3 and δ_4. There is a large amount of symmetry between these equations. For starters, the front steering angles are dependent on the front ICR radius and path angle, ρ_F and δ_F, whereas the rear steering angles are dependent on ρ_R and δ_R. Individually, each steering angle, δ_i, also depends on the magnitude of its' own ICR radius, ρ_i.

The triangles for solving the variables, ρ_F and ρ_R, are shown in both Figure 3.10 and Figure 3.11; these distances are determined using the Law of Sines as,

$$\rho_F = \frac{H\sin(\delta_R)}{\sin(\delta_F + \delta_R)}, \tag{3.41}$$

$$\rho_R = \frac{H\sin(\delta_F)}{\sin(\delta_F + \delta_R)}. \tag{3.42}$$

Equations (3.41) and (3.42) remove some of the unknowns from Equations (3.37)-(3.40). However, the individual ICR radius's, ρ_i, are still not solved for. While the Law-of-Sines cannot be used again, the Law-of-Cosines can be. Thus, the ICR radiuses for each steering angle turn out to be,

$$\rho_1^2 = {\rho_F}^2 + \left(\frac{W}{2}\right)^2 + \rho_F W cos(\delta_F), \tag{3.43}$$

$$\rho_2^2 = {\rho_F}^2 + \left(\frac{W}{2}\right)^2 - \rho_F W cos(\delta_F), \tag{3.44}$$

$$\rho_3^2 = {\rho_R}^2 + \left(\frac{W}{2}\right)^2 + \rho_R W cos(\delta_R), \tag{3.45}$$

$$\rho_4^2 = {\rho_R}^2 + \left(\frac{W}{2}\right)^2 - \rho_R W cos(\delta_R). \tag{3.46}$$

An additional equation can be made for ρ_B which is the ICR radius connecting the ICR to the COG,

$$\rho_B^2 = \frac{1}{2}{\rho_F}^2 + \frac{1}{2}{\rho_R}^2 - \left(\frac{H}{2}\right)^2. \tag{3.47}$$

Equation (3.43)-(3.46) provide the final unknowns to Equations (3.37)-(3.40). The same symmetry present in Equations (3.37)-(3.40) can be seen in Equation (3.43)-(3.46) as the front two ICR radius's are dependent on ρ_F and δ_F while the rear two are dependent on ρ_R and δ_R. However, Equations (3.41) and (3.42) show that ρ_F and ρ_R are dependent on their opposite path angle. That is, ρ_F is dependent on δ_R, and ρ_R is dependent on δ_F. This coupling, though unavoidable, will prove inconvenient in future simplifications. Equation (3.43)-(3.46) are also squared. The reasoning for this is that the time derivatives of the squared equations are simpler than those where the square root is taken. It is simple to also square Equations (3.37)-(3.40) and Equations (3.41) and (3.42) in order to accommodate the form of Equation (3.43)-(3.46).

The angle, δ_i, has been used to define the orientation of the wheel body in these derivations while Figure 3.2 and Figure 3.3 define the angle orientation using, Ψ_i. The reason for this change is that the angle, δ_i, is local, relative to the orientation of the robot, and depends on the position of the ICR whereas, Ψ_i, is global.

From here the generalized coordinate vector representing the robotic vehicle in its configuration space can be defined as,

$$q = [X_F \quad Y_F \quad X_R \quad Y_R \quad \delta_F \quad \delta_R]^T, \tag{3.48}$$

which defines the positions of all of the coordinates shown in Figure 3.3. For the sake of convenience for future mathematical operations, the vector can be redefined as,

$$q = [q_1 \quad q_2 \quad q_3 \quad q_4 \quad q_5 \quad q_6]^T. \tag{3.49}$$

3.5. Wheel Velocities and Body Yaw Rates

While the ICR has been useful for deriving the no-slip orientation, or steering angles, of the wheel bodies, it is also useful for determining the velocity for each respective wheel body. Figure 3.12 shows how the yaw-rate of the ICR is the same yaw-rate on the body.

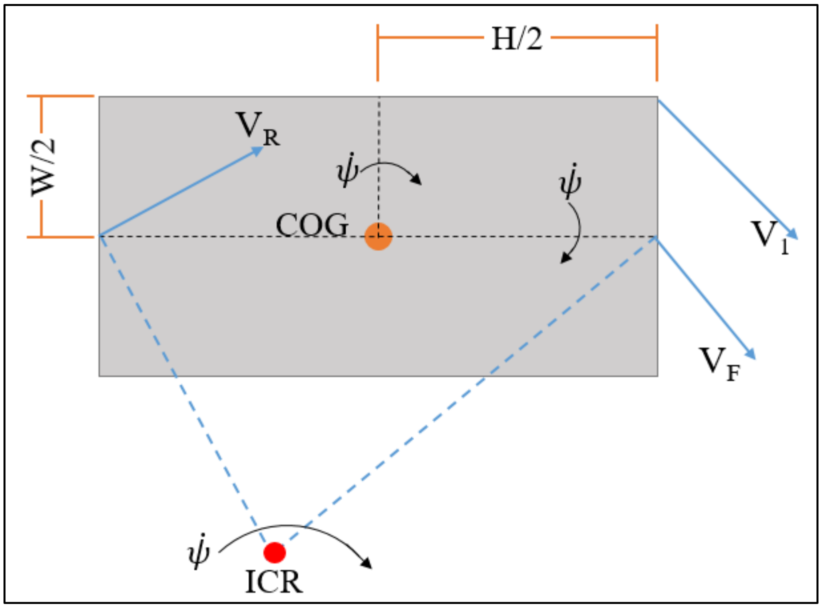

Figure 3.12: The relationship between the yaw-rate of a rigid body local velocity.

The velocity of any of the bodies can be determined with the x and y components, V_1, defined as the derivatives of the body positions of Equations (3.4)-(3.11) as

$$V_{1X} = V_B \cos(\psi_B) + \frac{H}{2}\dot{\psi} - \frac{W}{2}\dot{\psi}, \tag{3.50}$$

$$V_{1Y} = V_B \sin(\psi_B) + \frac{H}{2}\dot{\psi} + \frac{W}{2}\dot{\psi}. \tag{3.51}$$

Velocities defined this way requires the COG velocity and body yaw rates to be known. Figure 3.6 is very similar to Figure 3.12 in that both show the aspect ratio of the robot. The major difference is that Figure 3.6 focuses on the relationship between wheel body position and the path coordinates while Figure 3.12 shows the relationship between the path velocities, yaw-rate, and wheel body velocities. The wheel velocities can also be shown using path coordinates only. Thus, the components of the wheel velocities can be found through the time derivative of the wheel positions from Equations (3.15)-(3.21) without explicitly involving the body velocities and yaw rates. These velocity components are,

$$\dot{X}_1 = \dot{X}_F - \frac{W}{2H}(\dot{Y}_F - \dot{Y}_R), \tag{3.52}$$

$$\dot{Y}_1 = \dot{Y}_F + \frac{W}{2H}(\dot{X}_F - \dot{X}_R), \tag{3.53}$$

$$\dot{X}_2 = \dot{X}_F + \frac{W}{2H}(\dot{Y}_F - \dot{Y}_R), \tag{3.54}$$

$$\dot{Y}_2 = \dot{Y}_F - \frac{W}{2H}(\dot{X}_F - \dot{X}_R), \tag{3.55}$$

$$\dot{X}_3 = \dot{X}_R - \frac{W}{2H}(\dot{Y}_F - \dot{Y}_R), \tag{3.56}$$

$$\dot{Y}_3 = \dot{Y}_R + \frac{W}{2H}(\dot{X}_F - \dot{X}_R), \tag{3.57}$$

$$\dot{X}_4 = \dot{X}_R + \frac{W}{2H}(\dot{Y}_F - \dot{Y}_R), \tag{3.58}$$

$$\dot{Y}_4 = \dot{Y}_R - \frac{W}{2H}(\dot{X}_F - \dot{X}_R). \tag{3.59}$$

The chassis velocity can be expressed as,

$$\dot{X}_B = \frac{(\dot{X}_F + \dot{X}_R)}{2}, \tag{3.60}$$

$$\dot{Y}_B = \frac{(\dot{Y}_F + \dot{Y}_R)}{2}. \tag{3.61}$$

The squared magnitude of the wheel body or chassis velocities is the sum of the individual velocity components squared,

$$V_i^{\,2} = \left(\dot{X}_i^2 + \dot{Y}_i^2\right). \tag{3.62}$$

With this knowledge, the five body velocities become,

$$V_1^2 = \left(\dot{X}_F^2 + \dot{Y}_F^2\right) - \frac{W}{H}\dot{X}_F\left(\dot{Y}_F - \dot{Y}_R\right) + \frac{W}{H}\dot{Y}_F\left(\dot{X}_F - \dot{X}_R\right) + \frac{W^2}{4H^2}\left(\dot{X}_F^2 + \dot{X}_R^2 + \dot{Y}_F^2 + \dot{Y}_R^2 - 2\dot{X}_F\dot{X}_R - 2\dot{Y}_F\dot{Y}_R\right), \tag{3.63}$$

$$V_2^2 = \left(\dot{X}_F^2 + \dot{Y}_F^2\right) + \frac{W}{H}\dot{X}_F\left(\dot{Y}_F - \dot{Y}_R\right) - \frac{W}{H}\dot{Y}_F\left(\dot{X}_F - \dot{X}_R\right) + \frac{W^2}{4H^2}\left(\dot{X}_F^2 + \dot{X}_R^2 + \dot{Y}_F^2 + \dot{Y}_R^2 - 2\dot{X}_F\dot{X}_R - 2\dot{Y}_F\dot{Y}_R\right), \tag{3.64}$$

$$V_3^2 = \left(\dot{X}_R^2 + \dot{Y}_R^2\right) - \frac{W}{H}\dot{X}_R\left(\dot{Y}_F - \dot{Y}_R\right) + \frac{W}{H}\dot{Y}_R\left(\dot{X}_F - \dot{X}_R\right) + \frac{W^2}{4H^2}\left(\dot{X}_F^2 + \dot{X}_R^2 + \dot{Y}_F^2 + \dot{Y}_R^2 - 2\dot{X}_F\dot{X}_R - 2\dot{Y}_F\dot{Y}_R\right), \tag{3.65}$$

$$V_4^2 = \left(\dot{X}_R^2 + \dot{Y}_R^2\right) - \frac{W}{H}\dot{X}_R\left(\dot{Y}_F - \dot{Y}_R\right) + \frac{W}{H}\dot{Y}_R\left(\dot{X}_F - \dot{X}_R\right) + \frac{W^2}{4H^2}\left(\dot{X}_F^2 + \dot{X}_R^2 + \dot{Y}_F^2 + \dot{Y}_R^2 - 2\dot{X}_F\dot{X}_R - 2\dot{Y}_F\dot{Y}_R\right), \tag{3.66}$$

$$\dot{V}_B^2 = \frac{\left(\dot{X}_F^2 + \dot{X}_R^2 + \dot{Y}_F^2 + \dot{Y}_R^2 + 2\dot{X}_F\dot{X}_R + 2\dot{Y}_F\dot{Y}_R\right)}{4}. \tag{3.67}$$

Equation (3.3) shows the relationship between the path-coordinates and the orientation, yaw, of the robot. In order to derive the rate of change of orientation, yaw-rate, the time derivative of this must be taken,

$$sec^2(\psi_B)\dot{\psi}_B = \frac{(\dot{Y}_F - \dot{Y}_R)}{(X_F - X_R)} - \frac{(\dot{X}_F - \dot{X}_R)(Y_F - Y_R)}{(X_F - X_R)^2}. \tag{3.68}$$

Since cosine is defined in terms of the generalized coordinates in (3.13), the value for secant squared can be replaced with,

$$sec^2(\psi_B) = \frac{H^2}{(X_F - X_R)^2}. \tag{3.69}$$

Using Equations (3.68) and (3.69), the final form of the yaw-rate becomes,

$$\dot{\psi}_B = \frac{(\dot{Y}_F - \dot{Y}_R)(X_F - X_R) - (\dot{X}_F - \dot{X}_R)(Y_F - Y_R)}{H^2}. \tag{3.70}$$

3.6. Formulation of Constraints

While the robot motion can be described by six generalized coordinates, the robot body still has fifteen DOF. This means that nine constraint equations are needed so that the motion of every coordinate is defined at every moment.

Constraint equations have the form,

$$\varphi_i(q,t) = 0, \tag{3.71}$$

which simply means that there must be a function composed of the generalized coordinates that equates to zero.

These nine constraints can be broken into two categories: position constraints and velocity constraints. All constraints must maintain the form of Equation (3.71) in order to maintain their holonomy. To be able to include position constraints into the equations of motion, their time derivatives must be derived in velocity form for inclusion in the Lagrange equations of motion.

3.6.1. Position Constraints

The first constraint will have the form,

$$H^{method\ 1} - H = 0. \tag{3.72}$$

The term, H, in Equation (3.72) is the height of the robot, as shown in Figure 3.6. In order to incorporate, the following relationship can be utilized,

$$H^{method\ 1} = (X_F - X_R)^2 + (Y_F - Y_R)^2. \tag{3.73}$$

The constraint then takes the form of,

$$(X_F - X_R)^2 + (Y_F - Y_R)^2 - H^2 = 0. \tag{3.74}$$

The time derivative must be taken of Equation (3.74) such that it is of the form shown in Equation (3.71). Doing so leads to,

$$\left(\dot{X}_F - \dot{X}_R\right)(X_F - X_R) + \left(\dot{Y}_F - \dot{Y}_R\right)(Y_F - Y_R) = 0. \tag{3.75}$$

This can then be expressed using the generalized coordinates of Equation (3.49),

$$(q_1 - q_3)\dot{q}_1 + (q_2 - q_4)\dot{q}_2 - (q_1 - q_3)\dot{q}_3 - (q_2 - q_4)\dot{q}_4 = 0. \tag{3.76}$$

In order to increase the simplicity of the constraint matrix, a summation of the terms in Equation (3.76) leads to,

$$\sum_{k=1}^{6} C_{1,k}(q,t)\,\dot{q}_k = 0. \tag{3.77}$$

The form of Equation (3.77) is extremely valuable. Variable, C, represents the constraint matrix, which is a Jacobian of the constraints in Equation (3.73), while the subscripts, 1 and k, represent the constraint and term respectively. Since there will be nine constraints then there will be nine rows to the matrix. Similarly, each term in the columns represents the quantity which will be multiplied by the corresponding generalized coordinate velocity. That is, the first column corresponds to the first generalized coordinate velocity. With that, it can be seen that columns five and six of Equation (3.77) will be zero since that constraint does not depend on those velocities.

The second two position constraints are found using the Law-of-Sines. However, the location of the ICR must first be known before this trigonometric law can be used. A visual description of this method is presented in Figure 3.8.

Figure 3.10 and Figure 3.11 show the relationship between the path lengths, ρ_f and ρ_r, as well as the steering angles, δ_f and δ_r. Figure 3.8 provides a coordinate position, X_0 and Y_0, to the ICR in Figure 3.10 and Figure 3.11. The sine rule relationship can now be visualized as,

$$\frac{\left((X_F - X_0)^2 + (Y_F - Y_0)^2\right)}{\cos^2(\delta_R)} = \frac{\left((X_r - X_0)^2 + (Y_R - Y_0)^2\right)}{\cos^2(\delta_F)} = \frac{L^2}{\sin^2(\delta_F + \delta_R)}. \tag{3.78}$$

Therefore, the first constraint will have the form,

$$\frac{\left((X_F-X_0)^2+(Y_F-Y_0)^2\right)}{\cos^2(\delta_R)}-\frac{\left((X_r-X_0)^2+(Y_R-Y_0)^2\right)}{\cos^2(\delta_F)}=0, \tag{3.79}$$

while the second will have the form,

$$\frac{\left((X_r-X_0)^2+(Y_R-Y_0)^2\right)}{\cos^2(\delta_F)}-\frac{L^2}{\sin^2(\delta_F+\delta_R)}=0. \tag{3.80}$$

The time derivative leads to the new form of constraint two,

$$\left[\left(\dot{X}_F-\dot{X}_0\right)(X_F-X_0)+\left(\dot{Y}_F-\dot{Y}_0\right)(Y_F-Y_0)\right]\cos^2(\delta_F)-\dot{\delta}_F[(X_F-X_0)^2+(Y_F-Y_0)^2]\cos(\delta_F)\sin(\delta_F)-\left[\left(\dot{X}_R-\dot{X}_0\right)(X_R-X_0)+\left(\dot{Y}_R-\dot{Y}_0\right)(Y_R-Y_0)\right]\cos^2(\delta_R)+\dot{\delta}_R[(X_R-X_0)^2+(Y_R-Y_0)^2]\cos(\delta_R)\sin(\delta_R)=0. \tag{3.81}$$

It can be seen that the velocity of the ICR is needed in constraint two. In order to find the velocities, the ICR position from Equations (3.27) and (3.28), must be have their time derivatives taken.

The time derivative, expansion, and simplification of Equations (3.27) and (3.28) lead to the expressions,

$$\dot{X}_0=[1-\sin(q_5)\cos(q_6)\csc(q_5+q_6)\,\dot{q}_1]-\cos(q_5)\cos(q_6)\csc(q_5+q_6)\,\dot{q}_2+\sin(q_5)\cos(q_6)\csc(q_5+q_6)\,\dot{q}_3+\cos(q_5)\cos(q_6)\csc(q_5+q_6)\,\dot{q}_4+\{[(q_2-q_4)\sin(q_5)-(q_1-q_3)\cos(q_5)]\cos(q_6)\csc(q_5+q_6)-[(q_1-q_3)\sin(q_5)+(q_2-q_4)\cos(q_5)]\cos(q_6)\cot(q_5+q_6)\csc(q_5+q_6)\dot{q}_5\}-[(q_1-q_3)\sin(q_5)-(q_2-q_4)\cos(q_5)]\csc(q_5+q_6)\,(\sin(q_6)+\cos(q_6)\cot(q_5+q_6))\dot{q}_6, \tag{3.82}$$

$$\dot{Y}_0=[-\cos(q_5)\cos(q_6)\csc(q_5+q_6)\,\dot{q}_1]+[1+\cos(q_5)\cos(q_6)\csc(q_5+q_6)]\dot{q}_2+\sin(q_5)\cos(q_6)\csc(q_5+q_6)\,\dot{q}_3-\cos(q_5)\cos(q_6)\csc(q_5+q_6)\,\dot{q}_4+\{[(q_2-q_4)\sin(q_5)-(q_1-q_3)\cos(q_5)]\cos(q_6)\csc(q_5+q_6)-[(q_1-q_3)\sin(q_5)-(q_2-q_4)\cos(q_5)]\cos(q_6)\csc(q_5+q_6)\cot(q_5+q_6)\csc(q_5+q_6)\dot{q}_5\}-[(q_1-q_3)\sin(q_5)-(q_2-q_4)\cos(q_5)]\csc(q_5+q_6)\csc(q_5+q_6)\,(\sin(q_6)+\cos(q_6)\cot(q_5+q_6))\dot{q}_6. \tag{3.83}$$

The instant center vectorial velocities can be written as a matrix summation. The matrix will maintain the same form as the matrix, C, in Equation (3.77). However, it will have two columns, instead of nine, to accompany the two Equations: (3.82) and (3.83). It is written as,

$$\sum_{j=1}^{2} \sum_{k=1}^{6} H_{j,k}(q,t)\,\dot{q}_k = 0. \tag{3.84}$$

In matrix, H, the j column represents the equations for the X vector and the Y vector, in that order. Two more matrices can be created to further simplify the definition of constraints two and three. The matrix, D, will parse the position difference terms while the matrix, N, will parse the velocity difference terms. By doing this, constraints two and three will be of the form shown in Equation (3.77),

$$X_F - X_0 = [(q_1 - q_3)\sin(q_5) + (q_2 - q_4)\cos(q_5)]\cos(q_6)\csc(q_5 + q_6) = D_{1,1}(q,t), \tag{3.85}$$

$$Y_F - Y_0 = [(q_1 - q_3)\sin(q_5) - (q_2 - q_4)\cos(q_5)]\cos(q_6)\csc(q_5 + q_6) = D_{1,2}(q,t), \tag{3.86}$$

$$X_R - X_0 = [(q_1 - q_3)\cos(q_6) - (q_2 - q_4)\sin(q_6)]\cos(q_5)\csc(q_5 + q_6) = D_{2,1}(q,t), \tag{3.87}$$

$$Y_R - Y_0 = -[(q_1 - q_3)\cos(q_6) + (q_2 - q_4)\sin(q_6)]\cos(q_5)\csc(q_5 + q_6) = D_{2,2}(q,t). \tag{3.88}$$

The matrix, N, has a very similar form to the matrix, H. The first two rows are,

$$\dot{X}_F - \dot{X}_0 = \sin(q_5)\cos(q_6)\csc(q_5 + q_6)\,\dot{q}_1 + \cos(q_5)\cos(q_6)\csc(q_5 + q_6)\,\dot{q}_2 - \sin(q_5)\cos(q_6)\csc(q_5 + q_6)\,\dot{q}_3 - \cos(q_5)\cos(q_6)\csc(q_5 + q_6)\,\dot{q}_4 - \{[(q_2 - q_4)\sin(q_5) - (q_1 - q_3)\cos(q_5)]\cos(q_6)\csc(q_5 + q_6) - [(q_1 - q_3)\sin(q_5) + (q_2 - q_4)\cos(q_5)]\cos(q_6)\cot(q_5 + q_6)\csc(q_5 + q_6)\dot{q}_5\} - [(q_1 - q_3)\sin(q_5) + (q_2 - q_4)\cos(q_5)]\csc(q_5 + q_6)\,(\sin(q_6) + \cos(q_6)\cot(q_5 + q_6))\dot{q}_6 = \sum_{k=1}^{6} N_{1,k}(q,t)\dot{q}_k, \tag{3.89}$$

$$\dot{Y}_F - \dot{Y}_0 = \sin(q_5)\cos(q_6)\csc(q_5+q_6)\,\dot{q}_1 - \cos(q_5)\cos(q_6)\csc(q_5+q_6)\,\dot{q}_2 - \sin(q_5)\cos(q_6)\csc(q_5+q_6)\,\dot{q}_3 + \cos(q_5)\cos(q_6)\csc(q_5+q_6)\,\dot{q}_4 + \{[(q_2-q_4)\sin(q_5)+(q_1-q_3)\cos(q_5)]\cos(q_6)\csc(q_5+q_6) - [(q_1-q_3)\sin(q_5)+(q_2-q_4)\cos(q_5)]\cos(q_6)\csc(q_5+q_6)\cot(q_5+q_6)\csc(q_5+q_6)\dot{q}_5\} - [(q_1-q_3)\sin(q_5)-(q_2-q_4)\cos(q_5)]\csc(q_5+q_6)\csc(q_5+q_6)\,(\sin(q_6)+\cos(q_6)\cot(q_5+q_6))\dot{q}_6 = \sum_{k=1}^{6} N_{2,k}(q,t)\dot{q}_k, \tag{3.90}$$

$$\dot{X}_R - \dot{X}_0 = \cos(q_5)\cos(q_6)\csc(q_5+q_6)\,\dot{q}_1 - \cos(q_5)\sin(q_6)\csc(q_5+q_6)\,\dot{q}_2 - \cos(q_5)\cos(q_6)\csc(q_5+q_6)\,\dot{q}_3 + \cos(q_5)\sin(q_6)\csc(q_5+q_6)\,\dot{q}_4 - [(q_1-q_3)\cos(q_6)-(q_2-q_4)\sin(q_6)]\csc(q_5+q_6)\,(\sin(q_6)+\cos(q_5)\cot(q_5+q_6))\dot{q}_5 - \{[(q_1-q_3)\sin(q_6)+(q_2-q_4)\cos(q_6)]\cos(q_5)\csc(q_5+q_6) + [(q_1-q_3)\cos(q_6)-(q_2-q_4)\sin(q_6)]\cos(q_5)\csc(q_5+q_6)\cot(q_5+q_6)\}\dot{q}_6 = \sum_{k=1}^{6} N_{3,k}(q,t)\dot{q}_k, \tag{3.91}$$

$$\dot{Y}_R - \dot{Y}_0 = -\cos(q_5)\cos(q_6)\csc(q_5+q_6)\,\dot{q}_1 - \cos(q_5)\sin(q_6)\csc(q_5+q_6)\,\dot{q}_2 + \cos(q_5)\cos(q_6)\csc(q_5+q_6)\,\dot{q}_3 + \cos(q_5)\sin(q_6)\csc(q_5+q_6)\,\dot{q}_4 + [(q_1-q_3)\cos(q_6)+(q_2-q_4)\sin(q_6)]\csc(q_5+q_6)\,(\sin(q_6)+\cos(q_5)\cot(q_5+q_6))\dot{q}_5 + \{[(q_1-q_3)\sin(q_6)-(q_2-q_4)\cos(q_6)]\cos(q_5)\csc(q_5+q_6) + [(q_1-q_3)\cos(q_6)+(q_2-q_4)\sin(q_6)]\cos(q_5)\csc(q_5+q_6)\cot(q_5+q_6)\}\dot{q}_6 = \sum_{k=1}^{6} N_{4,k}(q,t)\dot{q}_k. \tag{3.92}$$

With these matrices defined, the first four column terms for the second and third constraints can be written in a more condensed form,

$$C_{2,k}(q,t) = \left[D_{1,1}(q,t)N_{1k}(q,t) + D_{1,2}(q,t)N_{2,k}(q,t)\right]\cos^2(q_5) - \left[D_{2,1}(q,t)N_{3,k}(q,t) + D_{2,2}(q,t)N_{4,k}(q,t)\right]\cos^2(q_6),\ k = 1{:}\,4, \tag{3.93}$$

$$C_{3,k}(q,t) = \left[D_{2,1}(q,t)N_{3,k}(q,t) + D_{2,2}(q,t)N_{4,k}(q,t)\right]\sin^2(q_5+q_6). \tag{3.94}$$

The fifth and sixth columns of the second and third constraint are more in-depth,

$$C_{2,5}(q,t) = \left[D_{,11}(q,t)N_{1,5}(q,t) + D_{1,2}(q,t)N_{2,5}(q,t)\right]\cos^2(q_5) - \left[D_{2,1}(q,t)N_{3,5}(q,t) + D_{2,2}(q,t)N_{4,5}(q,t)\right]\cos^2(q_6) - \left[D_{1,1}^2 + D_{1,2}^2\right]\cos(q_5)\sin(q_5), \tag{3.95}$$

$$C_{2,6}(q,t) = \left[D_{1,1}(q,t)N_{1,6}(q,t) + D_{1,2}(q,t)N_{2,6}(q,t)\right]\cos^2(q_5) - \left[D_{2,1}(q,t)N_{3,6}(q,t) + D_{2,2}(q,t)N_{4,k}(q,t)\right]\cos^2(q_6) + \left[D_{2,1}^2 + D_{2,2}^2\right]\cos(q_6)\sin(q_6), \tag{3.96}$$

$$C_{3,5}(q,t) = \left[D_{2,1}(q,t)N_{3,5}(q,t) + D_{2,2}(q,t)N_{4,5}(q,t)\right]\sin^2(q_5+q_6) - \left[D_{1,1}^2(q,t) + D_{1,2}^2(q,t)\right]\cos(q_5+q_6)\sin(q_5+q_6) + L^2\sin(q_5)\cos(q_5), \tag{3.97}$$

$$C_{3,6}(q,t) = \left[D_{2,1}(q,t)N_{3,6}(q,t) + D_{2,2}(q,t)N_{4,6}(q,t)\right]\sin^2(q_5+q_6) + \left[D_{1,1}^2(q,t) + D_{1,2}^2(q,t)\right]\cos(q_5+q_6)\sin(q_5+q_6). \tag{3.98}$$

Equations (3.93)-(3.98) give the coefficients for the second and third constraints. They can be written in the form,

$$\sum_{j=2}^{3}\sum_{k=1}^{6} C_{j,k}(q,t)\,\dot{q}_k = 0. \tag{3.99}$$

3.6.2. Velocity Constraints

Before the form of the wheel velocity constraints can be shown, the theory behind them must first be understood. For starts, the velocity of a point on a body can be related to the yaw-rate and the ICR radius,

$$V_i = \dot{\psi}_B \rho_i. \tag{3.100}$$

It was mentioned during the derivation of the wheel velocities that all the points on a rigid body have the same yaw-rate. This means that Equation (3.100) can be rearranged as,

$$\dot{\psi}_B = \frac{V_i}{\rho_i}, \tag{3.101}$$

and then rewritten as

$$\frac{V_j}{\rho_j} = \frac{V_i}{\rho_i},\ \ i \neq j. \tag{3.102}$$

A constraint equation can then be formed using the knowledge that a constraint takes the form of Equation (3.71). Six constraint equations can be made with the relationship shown in Equation (3.102),

$$\frac{V_1}{\rho_1} = \frac{V_2}{\rho_2} = \frac{V_3}{\rho_3} = \frac{V_4}{\rho_4} = \frac{V_F}{\rho_F} = \frac{V_R}{\rho_R} = \dot{\psi}_B. \tag{3.103}$$

It is important to not use redundant constraints as it will leave an element of the system unconstrained. This means there can be six constraint equations from the combination of the five velocities and ICR radius's, V_i and ρ_i. Seven constraint equations from these variables would result in a redundant constraint. However, the form of Equations (3.101) presents difficulties with the time derivative, as nonholonomy is introduced into the equations by the square present in Equations (3.63) through (3.67). For that reason, they must be modified. A velocity vector, has components in both the, i, and, j, unit vectors. Also, it is known that the magnitude of a vector can be written using sines and cosines. For example,

$$u_v = \cos(\theta)\,\bar{\imath} + \sin(\theta)\,\bar{\jmath}. \tag{3.104}$$

In Equation (3.104), u_v, represents the unit vector of an arbitrary vector while the angle, θ, represents the ratio of the vector in the specific coordinate direction. Thus, the velocity vector can be dotted with the unit vector to result in the magnitude of that vector,

$$\left(\dot{X}_j^2 + \dot{Y}_j^2\right)^{1/2} = \left(\dot{X}_j\bar{\imath} + \dot{Y}_j\bar{\jmath}\right)\cdot\left(\cos(\psi_j)\,\bar{\imath} + \sin(\psi_j)\,\bar{\jmath}\right) = \dot{X}_j\left(\frac{Y_j - Y_0}{\rho_j}\right) - \dot{Y}_j\left(\frac{X_j - X_0}{\rho_j}\right). \tag{3.105}$$

Simplification matrices, A, B, J, K, and G will be created to function much the same as matrices, H, D, and N. This will be done in order to simplify the velocity constraints. Some of these matrices will also be used in the kinetic energy section of the derivation.

The first of these matrices will be matrix, A, which defines the, X, component of the wheel linear velocities. The wheel linear velocities depend on their position relative to the path velocities, as illustrated in Figure 3.12. Thus, the individual, X component, wheel velocities can be redefined as,

$$\dot{X}_i = \left(\frac{1}{2} + \frac{X_i^{(0)}}{H}\right)\dot{q}_1 - \frac{Y_i^{(0)}}{H}\dot{q}_2 + \left(\frac{1}{2} - \frac{X_i^{(0)}}{H}\right)\dot{q}_3 + \frac{Y_i^{(0)}}{H}\dot{q}_4 = \sum_{i=1}^{4}\sum_{k=1}^{6} A_{i,k}(q,t)\,\dot{q}_k. \tag{3.106}$$

Equation (3.106) does not depend on the path angle rates, which means the terms, k=5 and k=6, are zero in the, A, matrix. The, B, matrix is very similar to the, A, matrix but focus on the Y component of the velocity,

$$\dot{Y}_i = \frac{Y_i^{(0)}}{H}\dot{q}_1 + \left(\frac{1}{2} + \frac{X_i^{(0)}}{H}\right)\dot{q}_2 - \frac{Y_i^{(0)}}{H}\dot{q}_3 + \left(\frac{1}{2} - \frac{X_i^{(0)}}{H}\right)\dot{q}_4 = \sum_{i=1}^{4}\sum_{k=1}^{6} B_{i,k}(q,t)\,\dot{q}_k. \tag{3.107}$$

The terms, $X_i^{(0)}$, and, $Y_i^{(0)}$, are the initial position of the Wheel-Body, i. This can be seen in Figure 3.6.

Assuming the COG of the robot starts at the coordinates, (0,0,0), the wheel positions from Equations (3.106) and (3.107) can be defined as,

$$X_1^{(0)} = \frac{H}{2}, X_2^{(0)} = \frac{H}{2}, X_3^{(0)} = -\frac{H}{2}, X_4^{(0)} = -\frac{H}{2}, \tag{3.108}$$

$$Y_1^{(0)} = \frac{W}{2}, Y_2^{(0)} = -\frac{W}{2}, Y_3^{(0)} = \frac{W}{2}, Y_4^{(0)} = -\frac{W}{2}. \tag{3.109}$$

The matrix, J, has many similarities to the matrix, D, except that it is the distance from the ICR to the wheel coordinates whereas, D, is the distance from the ICR to the path coordinates. J, is defined as,

$$X_0(q,t) - X_i(q,t) = q_1 - [(q_1 - q_3)\sin(q_5) + (q_2 - q_4)\cos(q_5)]\cos(q_6)\csc(q_5 + q_6) - \left(\frac{1}{2}(q_1 + q_3) + \frac{1}{L}\left[(q_1 - q_3)X_i^{(0)} - (q_2 - q_4)Y_i^{(0)}\right]\right) = J_{1,i}(q,t), \tag{3.110}$$

$$Y_0(q,t) - Y_i(q,t) = q_2 - [(q_1 - q_3)\sin(q_5) + (q_2 - q_4)\cos(q_5)]\cos(q_6)\csc(q_5 + q_6) - \left(\frac{1}{2}(q_1 + q_3) + \frac{1}{L}\left[(q_1 - q_3)X_i^{(0)} - (q_2 - q_4)Y_i^{(0)}\right]\right) = J_{2,i}(q,t). \tag{3.111}$$

With the matrices, A, B, and J defined, three of the six constraints, in the form of Equation (3.103) can be constructed,

$$\sum_{k=1}^{6}\left\{\frac{1}{\rho_1^2}\left[A_{1,k}(q,t)J_{1,1}(q,t) - B_{1,k}(q,t)J_{2,1}(q,t)\right] - \frac{1}{\rho_2^2}\left[A_{2,k}(q,t)J_{1,2}(q,t) - B_{2,k}(q,t)J_{2,2}(q,t)\right]\right\}\dot{q}_k = \sum_{k=1}^{6} C_{4,k}(q,t)\,\dot{q}_k, \tag{3.112}$$

$$\sum_{k=1}^{6}\left\{\frac{1}{\rho_2^2}\left[A_{2,k}(q,t)J_{1,2}(q,t) - B_{2,k}(q,t)J_{2,2}(q,t)\right] - \frac{1}{\rho_3^2}\left[A_{3,k}(q,t)J_{1,3}(q,t) - B_{3,k}(q,t)J_{2,3}(q,t)\right]\right\}\dot{q}_k = \sum_{k=1}^{6} C_{5,k}(q,t)\,\dot{q}_k, \tag{3.113}$$

$$\sum_{k=1}^{6}\left\{\frac{1}{\rho_3^2}\left[A_{3,k}(q,t)J_{1,3}(q,t) - B_{3,k}(q,t)J_{2,3}(q,t)\right] - \frac{1}{\rho_4^2}\left[A_{4,k}(q,t)J_{1,4}(q,t) - B_{4,k}(q,t)J_{2,4}(q,t)\right]\right\}\dot{q}_k = \sum_{k=1}^{6} C_{6,k}(q,t)\,\dot{q}_k. \tag{3.114}$$

The seventh constraint uses the path coordinate velocities in its derivation. Using the relationship shown in Equation (3.104), the following expression can be created,

$$\left(\dot{X}_F^2 + \dot{Y}_F^2\right)^{1/2} = \left(\dot{X}_F\bar{\imath} + \dot{Y}_F\bar{\jmath}\right)\cdot\left(\cos(\psi_B+\delta_F)\,\bar{\imath} + \sin(\psi_B+\delta_F)\,\bar{\jmath}\right) = \dot{X}_F\cos(\psi_B + \delta_F) + \dot{Y}_j\sin(\psi_B+\delta_F). \tag{3.115}$$

Using the relationships for Ψ_B, rearranging terms, and simplifying results in an equivalent expression to Equation (3.115),

$$\left(\dot{X}_F^2 + \dot{Y}_F^2\right)^{1/2} = \frac{1}{L}\left[(q_1-q_3)\cos(q_5) - (q_2-q_4)\sin(q_5)\right]\dot{q}_1 + \frac{1}{L}\left[(q_2 - q_4)\cos(q_5) + (q_1-q_3)\sin(q_5)\right]\dot{q}_2 = \sum_{k=1}^{6} K_{1,k}(q,t)\,\dot{q}_k. \tag{3.116}$$

A similar relationship can be made for the rear path coordinate,

$$\left(\dot{X}_R^2 + \dot{Y}_R^2\right)^{1/2} = \frac{1}{L}\left[(q_1-q_3)\cos(q_6) + (q_2-q_4)\sin(q_6)\right]\dot{q}_1 + \frac{1}{L}\left[(q_2 - q_4)\cos(q_6) - (q_1-q_3)\sin(q_6)\right]\dot{q}_2 = \sum_{k=1}^{6} K_{2,k}(q,t)\,\dot{q}_k. \tag{3.117}$$

Using the matrices, A, B, J, and K, the seventh and eight constraints are,

$$\sum_{k=1}^{6}\left\{\frac{1}{\rho_4^2}\left[A_{4,k}(q,t)J_{1,4}(q,t) - B_{4,k}(q,t)J_{1,4}(q,t)\right] - \frac{1}{\rho_F}K_{1,k}(q,t)\right\}\dot{q}_k = \sum_{k=1}^{6} C_{7,k}(q,t)\,\dot{q}_k, \tag{3.118}$$

$$\sum_{k=1}^{6}\left\{\frac{1}{\rho_F}K_{1,k}(q,t) - \frac{1}{\rho_r}K_{2,k}(q,t)\right\}\dot{q}_k = \sum_{k=1}^{6} C_{8,k}(q,t)\,\dot{q}_k. \tag{3.119}$$

To define the ninth and final constraint, one more simplification matrix is needed. This matrix, G, will define the body velocities shown in Equations (3.60), (3.61), and (3.70). This matrix is,

$$\dot{X}_B = \frac{1}{2}(\dot{q}_1 + \dot{q}_3) = \sum_{k=1}^{6} G_{1,k}(q,t)\,\dot{q}_k, \tag{3.120}$$

$$\dot{Y}_B = \frac{1}{2}(\dot{q}_2 + \dot{q}_4) = \sum_{k=1}^{6} G_{2,k}(q,t)\,\dot{q}_k, \tag{3.121}$$

$$\dot{\psi}_B = -\frac{1}{L^2}(q_2 - q_4)\dot{q}_1 + \frac{1}{L^2}(q_1 - q_3)\dot{q}_2 + \frac{1}{L^2}(q_2 - q_4)\dot{q}_3 - \frac{1}{L^2}(q_1 - q_3)\dot{q}_4 = \sum_{k=1}^{6} G_{3,k}(q,t)\,\dot{q}_k. \tag{3.122}$$

Using the matrices, K and G, the final constraint can be shown as,

$$\sum_{k=1}^{6}\left\{\frac{1}{\rho_r}K_{2,k}(q,t) - G_{3,k}(q,t)\right\}\dot{q}_k = \sum_{k=1}^{6} C_{9,k}(q,t)\,\dot{q}_k. \tag{3.123}$$

The complete set of nine constraints can be expressed in holonomic form as,

$$f(q,\dot{q},t) = \begin{bmatrix} \sum_{k=1}^{6} C_{1,k}(q,t)\,\dot{q}_k \\ \sum_{k=1}^{6} C_{2,k}(q,t)\,\dot{q}_k \\ \sum_{k=1}^{6} C_{3,k}(q,t)\,\dot{q}_k \\ \sum_{k=1}^{6} C_{4,k}(q,t)\,\dot{q}_k \\ \sum_{k=1}^{6} C_{5,k}(q,t)\,\dot{q}_k \\ \sum_{k=1}^{6} C_{6,k}(q,t)\,\dot{q}_k \\ \sum_{k=1}^{6} C_{7,k}(q,t)\,\dot{q}_k \\ \sum_{k=1}^{6} C_{8,k}(q,t)\,\dot{q}_k \\ \sum_{k=1}^{6} C_{9,k}(q,t)\,\dot{q}_k \end{bmatrix} = 0. \tag{3.124}$$

3.7. Formulation of the Generalized Forces

The wheels, due to a torque from the DC motors, exert a tractive force on the surface of motion. The direction of this force will be a vector in the global X-Y plane. The steering motors produce a torque, in the z-axis. These forces and torques can be seen in Figure 3.13 where the tractive force of each wheel is represented by, F_{Ti}, while the steering torques are represented by, T_{si}.

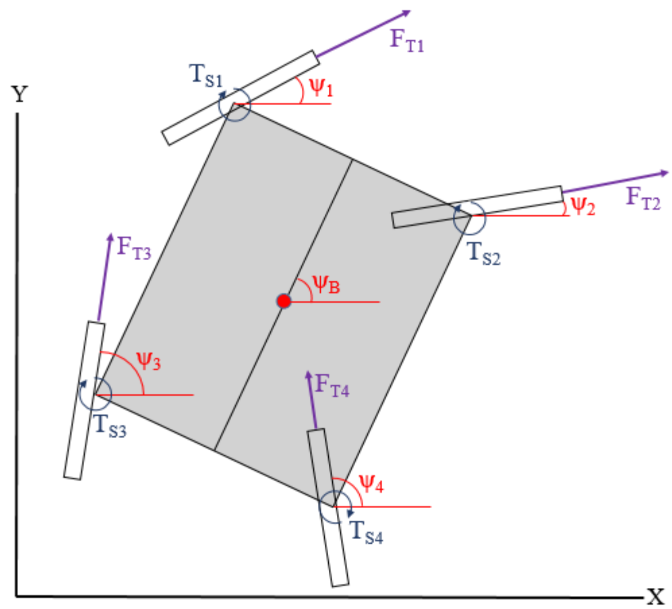

Figure 3.13: The forces and torques exerted on the ground by the multi-body system. F_{Ti} represents the tractive forces exerted by the wheels on the ground while T_{Si} is the torques exerted on the ground by the steering motors.

The vectorial tractive effort, F_{Ti}, can be written as,

$$\overline{F_{T1}} = F_{T1}(\cos(\psi_1)\,\bar{\imath} + \sin(\psi_1)\,\bar{\jmath}), \tag{3.125}$$

$$\overline{F_{T2}} = F_{T2}(\cos(\psi_2)\,\bar{\imath} + \sin(\psi_2)\,\bar{\jmath}), \tag{3.126}$$

$$\overline{F_{T3}} = F_{T3}(\cos(\psi_3)\,\bar{\imath} + \sin(\psi_3)\,\bar{\jmath}), \tag{3.127}$$

$$\overline{F_{T4}} = F_{T4}(\cos(\psi_4)\,\bar{\imath} + \sin(\psi_4)\,\bar{\jmath}). \tag{3.128}$$

However, the relationship formed in Equations (3.29)-(3.32) must be used to put the forces in terms of more convenient coordinates. Though the body orientation can be determined based off of the position of the path coordinates, as shown in Equation (3.3), it is often more convenient to use the IMU measured yaw angle in conjunction with the global position determined from the GPS to determine the robot's pose. Additionally, the steering angles are relative to the body which makes the global angle a simple summation of the body yaw and the steering angle.

The tractive effort exerted on the ground is higher than the actual force absorbed by the ground. This is because the wheel bodies and chassis have the same acceleration. Since the accelerations are the same then the distribution of forces cannot be equal as the masses are not equal. Thus, the tractive effort, F_{Ti}, must me multiplied by some factor to describe the amount of force absorbed by the ground and the body,

$$\overline{R_\iota}(q,t) = \gamma_i \overline{F_{T\iota}}, \tag{3.129}$$

where, γ_i, is the scaling factor between the tractive effort and the tractive force. From here the tractive forces can be summed to be expressed as,

$$\overline{F_B}(q,t) = \sum_{i-1}^{4} \overline{R_\iota}(q,t). \tag{3.130}$$

The moment reaction at the body is the result of the individual wheel forces in conjunction with their location relative to the body. Figure 3.14 shows how how the position of the wheels are defined relative to the COG. While this has already been discussed, primariliy in Equations (3.4) through (3.11), it can sometimes be easier to represent the positions in vectorial format. Thus, the moment reaction is,

$$\overline{M_B}(q,t) = \sum_{i-1}^{4} \left[\bar{r}_{i/G}(q,t) - \bar{r}_{B/G}(q,t)\right] \text{x}\, \overline{R_\iota}(q,t). \tag{3.131}$$

The wheel torques can be written as,

$$\overline{T_{S\iota}} = T_{Si}\bar{k}. \tag{3.132}$$

The resultant force and moment at the wheels can be expressed as,

$$\bar{F}_i(q,t) = (1-\gamma_i)\overline{F_{T\iota}}\bar{k}, \tag{3.133}$$

$$\overline{M_\iota}(q,t) = \overline{T_{S\iota}}. \tag{3.134}$$

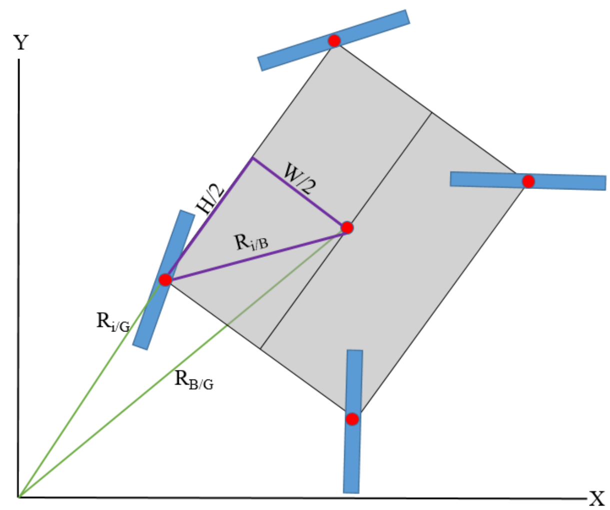

Figure 3.14: The position of the COG, in the global frame, is shown by RB/G. The position of an arbitrary wheel-body, in the global frame, is shown by Ri/G. The position of Ri/G can be related to RB/G with the dimensions, W and H, of the robot.

The generalized forces for the system can be expressed as,

$$Q_j = \frac{\partial \bar{r}_{B/G}}{\partial q_j} \cdot \overline{F_B}(q,t) + \frac{\partial \psi_B(q,t)\bar{k}}{\partial q_j}\overline{M_B}(q,t) + \sum_{i=1}^{4}\left[\frac{\partial \bar{r}_{i/G}}{\partial q_j} \cdot \overline{F_\iota}(q,t) + \frac{\partial \psi_i(q,t)\bar{k}}{\partial q_j}\overline{M_\iota}(q,t)\right]. \tag{3.135}$$

Equation (3.135) is very complex. However, steps can be taken to decrease the complexity to a small degree. Using Equations (3.4) through (3.11) can be used to create the vectors, $r_{i/G}$. With that, the following derivatives can be created,

$$\frac{\partial \bar{r}_{i/G}}{\partial q_1} = \left(\frac{1}{2} + \frac{X_i^0}{L}\right)\bar{\iota} + \left(\frac{Y_i^0}{L}\right)\bar{\jmath}, \tag{3.136}$$

$$\frac{\partial \bar{r}_{i/G}}{\partial q_2} = -\left(\frac{Y_i^0}{L}\right)\bar{\iota} + \left(\frac{1}{2} + \frac{X_i^0}{L}\right)\bar{\jmath}, \tag{3.137}$$

$$\frac{\partial \bar{r}_{i/G}}{\partial q_3} = \left(\frac{1}{2} - \frac{X_i^0}{L}\right)\bar{\iota} - \left(\frac{Y_i^0}{L}\right)\bar{\jmath}, \tag{3.138}$$

$$\frac{\partial \bar{r}_{i/G}}{\partial q_4} = \left(\frac{Y_i^0}{L}\right)\bar{\imath} + \left(\frac{1}{2} - \frac{X_i^0}{L}\right)\bar{J}. \tag{3.139}$$

Also, the following relationship is known,

$$\frac{\partial \psi_i}{\partial q_j} = \frac{1}{\rho_i^2}\left[J_{1,i}\frac{\partial J_{2,i}}{\partial q_j} - J_{2,i}\frac{\partial J_{1,i}}{\partial q_j}\right]. \tag{3.140}$$

The final generalized force equations can be written as,

$$Q_1 = \sum_{i=1}^{4}\Bigg\{F_{Ti}\left[\frac{1}{2} + (1-\gamma_i)\frac{X_i^0}{L} + \left(\frac{q_2-q_4}{L^3}\right)\gamma_i[(q_2-q_4)X_i^0 + (q_1-q_3)Y_i^0]\right]\left(\frac{-J_{2,i}}{\rho_i}\right) + F_{Ti}\left[(1-\gamma_i)\frac{Y_i^0}{L} - \left(\frac{q_2-q_4}{L^3}\right)(\gamma_i)[(q_1-q_3)X_i^0 - (q_2-q_4)Y_i^0]\right]\left(\frac{J_{1,i}}{\rho_i}\right) + \frac{T_{si}}{\rho_i^2}\left[J_{1,i}\left[-\sin(q_5)\cos(q_6)\csc(q_5+q_5) - \frac{Y_i^0}{L}\right] - J_{2,i}\left[1 - \left(\sin(q_5)\cos(q_5)\csc(q_5+q_6) - \frac{1}{2} - \frac{X_i^0}{L}\right)\right]\right]\Bigg\}, \tag{3.141}$$

$$Q_2 = \sum_{i=1}^{4}\Bigg\{F_{Ti}\left[-(1-\gamma_i)\frac{Y_i^0}{L} - \left(\frac{q_1-q_3}{L^3}\right)\gamma_i[(q_2-q_4)X_i^0 + (q_1-q_3)Y_i^0]\right]\left(\frac{-J_{2,i}}{\rho_i}\right) + F_{Ti}\left[\frac{1}{2} + (1-\gamma_i)\frac{X_i^0}{L} + \left(\frac{q_1-q_3}{L^3}\right)(\gamma_i)[(q_1-q_3)X_i^0 - (q_2-q_4)Y_i^0]\right]\left(\frac{J_{1,i}}{\rho_i}\right) + \frac{T_{si}}{\rho_i^2}\left[J_{1,i}\left[1 + \cos(q_5)\cos(q_6)\csc(q_5+q_5) - \frac{1}{2} - \frac{X_i^0}{L}\right] - J_{2,i}\left[-\left(\cos(q_5)\cos(q_5)\csc(q_5+q_6) + \frac{Y_i^0}{L}\right)\right]\right]\Bigg\}, \tag{3.142}$$

$$Q_3 = \sum_{i=1}^{4}\Bigg\{F_{Ti}\left[\frac{1}{2} - (1-\gamma_i)\frac{X_i^0}{L} - \left(\frac{q_2-q_4}{L^3}\right)\gamma_i[(q_2-q_4)X_i^0 + (q_1-q_3)Y_i^0]\right]\left(\frac{-J_{2,i}}{\rho_i}\right) + F_{Ti}\left[(1-\gamma_i)\frac{Y_i^0}{L} - \left(\frac{q_2-q_4}{L^3}\right)(\gamma_i)[(q_1-q_3)X_i^0 - (q_2-q_4)Y_i^0]\right]\left(\frac{J_{1,i}}{\rho_i}\right) - \frac{T_{si}}{\rho_i^2}\left[J_{1,i}\left[\sin(q_5)\cos(q_6)\csc(q_5+q_5) + \frac{Y_i^0}{L}\right] - J_{2,i}\left[\sin(q_5)\cos(q_5)\csc(q_5+q_6) - \frac{1}{2} + \frac{X_i^0}{L}\right]\right]\Bigg\}, \tag{3.143}$$

$$Q_4 = \sum_{i=1}^{4} \left\{ F_{Ti} \left[(1-\gamma_i) \frac{Y_i^0}{L} + \left(\frac{q_1 - q_3}{L^3} \right) \gamma_i [(q_2 - q_4) X_i^0 + (q_1 - q_3) Y_i^0] \right] \left(\frac{-J_{2,i}}{\rho_i} \right) + F_{Ti} \left[\frac{1}{2} - (1-\gamma_i) \frac{X_i^0}{L} - \left(\frac{q_1 - q_3}{L^3} \right) (\gamma_i) [(q_1 - q_3) X_i^0 - (q_2 - q_4) Y_i^0] \right] \left(\frac{J_{1,i}}{\rho_i} \right) + \frac{T_{si}}{\rho_i^2} \left[J_{1,i} \left[-\cos(q_5)\cos(q_6)\csc(q_5 + q_5) - \frac{1}{2} + \frac{X_i^0}{L} \right] - J_{2,i} \left[\cos(q_5)\cos(q_5)\csc(q_5 + q_6) - \frac{Y_i^0}{L} \right] \right] \right\}, \tag{3.144}$$

$$Q_5 = \sum_{i=1}^{4} \frac{T_{si}}{\rho_i^2} \csc(q_5 + q_5) \{ J_{1,i} [[-(q_1 - q_3)\cos(q_5) + (q_2 - q_4)\sin(q_5)] \cos(q_6) + [(q_1 - q_3)\sin(q_5) - (q_2 - q_4)\cos(q_5)] \cos(q_6)\cot(q_5 + q_6)] - J_{2,i} [-[(q_1 - q_3)\cos(q_5) - (q_2 - q_4)\sin(q_5)] \cos(q_6) + [(q_1 - q_3)\sin(q_5) + (q_2 - q_4)\cos(q_5)] \cos(q_6)\cot(q_5 + q_6)] \}, \tag{3.145}$$

$$Q_6 = \sum_{i=1}^{4} \frac{T_{si}}{\rho_i^2} \csc(q_5 + q_5) \{ J_{1,i} [[(q_1 - q_3)\sin(q_5) - (q_2 - q_4)\cos(q_5)] (\sin(q_6) + \cos(q_6)\cot(q_5 + q_6))] - J_{2,i} [[(q_1 - q_3)\sin(q_5) + (q_2 - q_4)\cos(q_5)] (\sin(q_6) + \cos(q_6)\cot(q_5 + q_6))] \}. \tag{3.146}$$

3.8. Formulation of the Lagrangian of Motion

The Lagrangian of the multi-body system is a sum of the individual kinetic energies of the system minus the system's potential energies,

$$L = \Sigma T - \Sigma V. \tag{3.147}$$

In Equation (3.147), L, is the Lagragian, T, is the kinetic energy, and, V, is the potential energy. Since there are no springs, and since the robot is constrained to a level surface where there are no changes in elevation, there is no change in potential energy. The potential energy can be eliminated by setting the reference for the potential energy to the COG of the robot. The Lagrangian then depends entirely on the kinetic energy since the potential energy is zero.

The Equations of Motion (EOM) for the system can be found using the Lagrangian Derivative,

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = Q + \lambda C(\dot{q}, q). \tag{3.148}$$

In Equation (3.148), Q is the generalized force vector, λ is the LaGrange multiplier, and C are the kinematic constraints. L, from Equation (3.147), is the Lagrangian and, q, from Equation (3.48) is the generalized coordinate vector, since the robot is composed of rigid bodies, as opposed to particles, the corresponding kinetic energy of each of the bodies will be composed of both translational and rotational components. The bodies are shown in Figure 3.2. The total kinetic energy for the system is,

$$T = \mathrm{T}_{B_{translation}} + \mathrm{T}_{B_{rotation}} + \sum_{i=1}^{4} \mathrm{T}_{i_{translation}} + \mathrm{T}_{i_{steering}} + \mathrm{T}_{i_{drive}}. \tag{3.149}$$

Each kinetic energy component, from Equation (3.149) can be written in terms of the generalized coordinates. The body has two components of motion. These can be combined into one expression,

$$T_{B_{translation}} + T_{B_{rotation}} = \frac{1}{2} m_B\left(\dot{X}_B^2 + \dot{Y}_B^2\right) + \frac{1}{2} I_B\left(\dot{\psi}_B\right)^2. \tag{3.150}$$

The terms in Equation (3.150) can be expressed in terms of the matrix, G, shown in Equations (3.120), (3.121), and (3.122),

$$T_B = \frac{1}{2} m_B \sum_{j=1}^{6} \sum_{k=1}^{6} \left[G_{1,j}(q,t) G_{1,k}(q,t) + G_{2,j}(q,t) G_{2,k}(q,t)\right] \dot{q}_j \dot{q}_k + \frac{1}{2} I_B \sum_{j=1}^{6} \sum_{k=1}^{6} \left[G_{3,j}(q,t) G_{3,k}(q,t)\right] \dot{q}_j \dot{q}_k. \tag{3.151}$$

The components of the kinetic energy for the wheels is expressed in three distinct terms, as shown in Equation (3.149). However, the kinetic energy created from the rotational inertia of the wheels can be directly related to the translational velocity of the frame. That is the wheel positions shown in Figure 3.6 and velocity in Figure 3.12 show that the translational velocity can be derived from the velocity of the body. While this is true, it is known that the physical system works in reverse as the body movement is derived from the wheel motion. This means the linear distance traveled by the wheels is equivalent to the translational movement of the frame at the location of each wheel-body. Assuming there is no slip in the system, this relationship can be shown as,

$$\omega_i = \frac{V_i}{r_w}. \tag{3.152}$$

In Equation (3.152), the wheel radius is represented by the variable, r_w, and the angular velocity is shown as, ω_i. Then, two of the three wheel kinetic energy terms, not including the steering, from Equation (3.149) can be summed into one equation,

$$\mathrm{T}_{i_{translation}} + \mathrm{T}_{i_{drive}} = \frac{1}{2}\left(m_w + \frac{I_w}{r_w^2}\right)\left(\dot{X}_i^2 + \dot{Y}_i^2\right). \tag{3.153}$$

Equation (3.106) and (3.107) already show the wheel velocities in matrix form with the, X, velocities in matrix, A, and the, Y, velocities in matrix, B. With that, Equation (3.153) can be represented as the following summation,

$$T_i = \frac{1}{2}\left(m_w + \frac{I_w}{r_w^2}\right)\sum_{i=1}^{4}\sum_{j=1}^{6}\sum_{k=1}^{6}\left[A_{i,j}(q,t)A_{i,k}(q,t) + B_{i,j}(q,t)B_{i,k}(q,t)\right]\dot{q}_j\dot{q}_k. \tag{3.154}$$

The term, $T_{isteering}$, from Equation (3.149) is the most complex term in the Lagrangian as it is based off the yaw-rate of the individual steering angles. The derivation of the steering angles can be seen in Equations (3.37) through (3.40). However, these angles are relative to the robot. This means that the derivative of these equations would be the angular rate relative to the robot. This means that using these equations would result in an inaccurate Lagrangian. For that reason, the global yaw-rates, shown in Equations (3.29) through (3.32), must be used in the Lagrangian,

$$T_i = \frac{1}{2}(I_S)\left(\dot{\psi}_i^2\right). \tag{3.155}$$

While Equations (3.29) through (3.32) show the global steering yaw angles as a function of the body position and the local steering angles. However, these can be written in a less complicated form,

$$\psi_i = \tan^{-1}\left(\frac{X_0(q,t)-X_i(q,t)}{Y_i(q,t)-Y_0(q,t)}\right) = \tan^{-1}\left(\frac{-J_{1,i}(q,t)}{J_{2,i}(q,t)}\right). \tag{3.156}$$

The time derivative of the global wheel yaw angles can then be written as,

$$\dot{\psi}_i = \frac{\left(Y_i(q,t)-Y_0(q,t)\right)\left(\dot{X}_0(q,t)-\dot{X}_i(q,t)\right)-\left(X_0(q,t)-X_i(q,t)\right)\left(\dot{Y}_i(q,t)-\dot{Y}_0(q,t)\right)}{\left(X_0(q,t)-X_i(q,t)\right)^2+\left(Y_i(q,t)-Y_0(q,t)\right)^2}. \tag{3.157}$$

The global wheel yaw rates can be written in matrix form such that they are not only expressed in generalized coordinates, but also simplified in their presentation,

$$\dot{\psi}_i = \frac{1}{\rho_i^2}\sum_{k=1}^{6}\left\{J_{2,i}(q,t)\left[A_{i,k}(q,t) - H_{1,k}(q,t)\right] - J_{1,i}(q,t)\left[B_{i,k}(q,t) - H_{2,k}(q,t)\right]\right\}\dot{q}_k = \sum_{k=1}^{6} S_{i,k}(q,t)\,\dot{q}_k. \tag{3.158}$$

With the global yaw-rates now defined, Equation (3.157) can be redefined as,

$$T_{isteering} = \frac{1}{2} I_S \sum_{i=1}^{4}\sum_{j=1}^{6}\sum_{k=1}^{6} S_{i,j}(q,t) S_{i,k}(q,t)\,\dot{q}_j\dot{q}_k. \tag{3.159}$$

The corresponding Euler-Lagrange equations of motion, derived from Equation (3.148), are

$$\sum_{k=1}^{6}\Big\{m_B\left[G_{1,j}(q,t)G_{1,k}(q,t) - G_{2,j}(q,t)G_{2,k}(q,t)\right] + I_B G_{3,j}(q,t)G_{3,k}(q,t) + \sum_{i=1}^{4}\left(m_w + \frac{I_w}{r_w^2}\right)\left[A_{i,j}(q,t)A_{i,k}(q,t) - B_{i,j}(q,t)B_{i,k}(q,t)\right] + I_s S_{i,j}(q,t)S_{i,k}(q,t)\Big\}\ddot{q}_k + I_S\sum_{k=1}^{6}\sum_{n=1}^{6}\sum_{i=1}^{4}\left(S_{i,j}^T(q,t)\frac{\partial S_{i,k}(q,t)}{\partial q_n} + S_{i,k}^T(q,t)\frac{\partial S_{i,j}(q,t)}{\partial q_n}\right)\dot{q}_n\dot{q}_k - \frac{1}{2} I_S\sum_{k=1}^{6}\sum_{n=1}^{6}\sum_{i=1}^{4}\left(S_{i,j}^T(q,t)\frac{\partial S_{i,k}(q,t)}{\partial q_j} + S_{i,k}^T(q,t)\frac{\partial S_{i,n}(q,t)}{\partial q_j}\right)\dot{q}_n\dot{q}_k = Q_j(q,t) + \sum_{k=1}^{9}\lambda_k\, C_{k,j}(q,t), \tag{3.160}$$

where λ_k are Lagrange multipliers. The simplified expression turns out to be due to derivatives for many of the terms die out in the mass matrix and are zero as a result; they can be expressed in a condensed form as

$$\sum_{k=1}^{6}\left[\Theta_{j,k}(q,t)\ddot{q}_k + \sum_{n=1}^{6}\Psi_{j,k,n}(q,t)\dot{q}_n\dot{q}_k\right] = Q_j(q,t) + \sum_{k=1}^{9}\lambda_k\, C_{k,j}(q,t). \tag{3.161}$$

In Equation (3.161), $\Theta_{j,k}$, is the generalized mass matrix and, $\Psi_{j,k,n}$, are the generalized damping coefficients.

4. NUMERIC SIMULATION

Computer simulations were done in order to verify the response of the created model. The vehicle Parameters are shown in Table 1. These values are used from, [37]. The code used for the simulation can be seen in 0.

Table 4.1: The numeric constants used for the simulations

M_B	50 (kg)
I_B	5.5 (kg/m^2)
Mi	3.5 (kg)
I_i	.025 (kg/m^2)
I_s	.009 (kg/m^2)
W	.75 (m)
H	1 (m)
r_w	.085 (m)

The model is extremely nonlinear, as shown in the derivation from Chapter 3. However, simple simulations can still be performed, in lieu of a controller, in order to see if the system's response is realistic. Figure 4.1 shows the velocity response of the robot undergoing a turn maneuver. It can be seen that the wheels accelerate at a constant velocity, which makes sense considering the force applied is constant. Also, it can be seen that the pairs of wheels, one and three as well as two and four, are inverted. This makes sense since those pairs are opposites, as shown in Figure 3.3. The position of wheel two goes negative while wheel four goes positive. This means the velocity of two should be negative while four is positive.

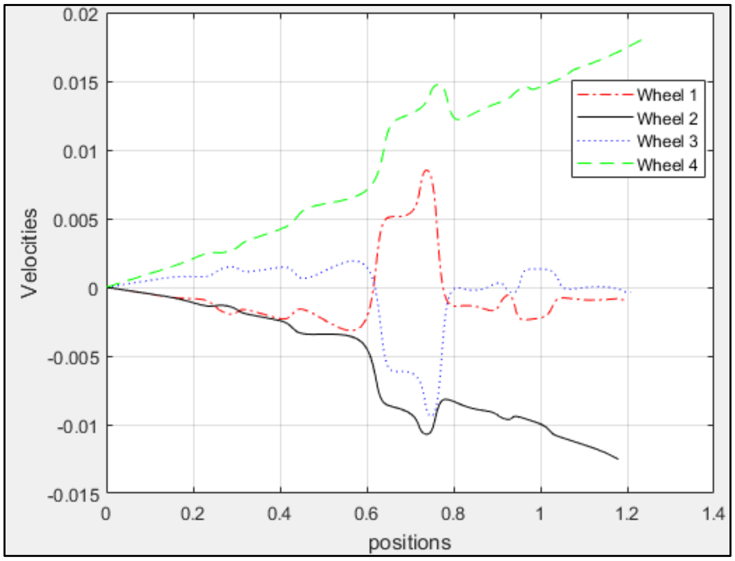

Figure 4.1: The velocities vs positions using a constant force input

Figure 4.2 shows the front and rear path coordinates on straight-line trajectory. The track is created by using steering angles of zero and constant force applied to the drive wheels which means straight line motion should be produced. The coordinates almost overlap perfectly.

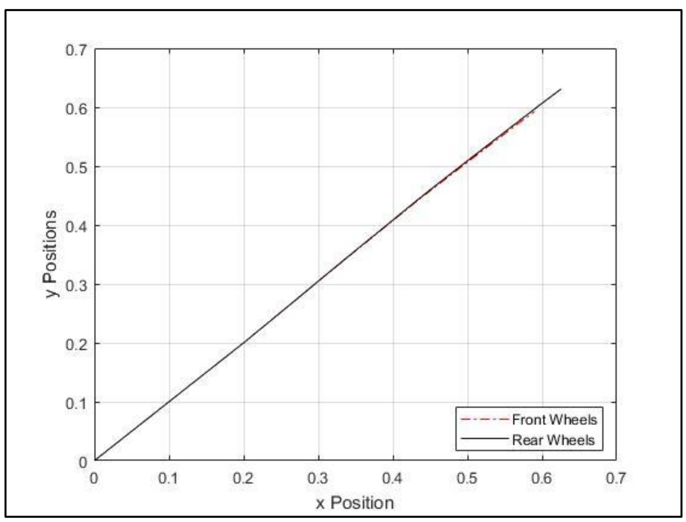

Figure 4.2: Front and Rear Coordinates on a straight line trajectory

The simulations from Figure 4.1 and Figure 4.2 both show model accuracy, albeit different facets of it. The accuracy displayed in Figure 4.1 is that of the wheels and how they respond to force and torque inputs while the accuracy of Figure 4.2 is for the body to see if it consistently follows a straight line given inputs for that straight line.

There system is inherently unstable due to the nonlinear nature of the model. Another part of the instability is most likely due to the poor integration from the trapezoidal integrator used in the model. An accurate integrator, such as fourth order Runge-Kutta, as well as a controller would most likely clean up the response from Figure 4.1.

5. EXPERIMENTAL SIMULATION

While the experimental system was created, there was not time for experimental testing. All the sensors and actuators are operational, however, the overall programming architecture of the system was not completed. Further work needed to be done in several areas, including, increasing communication reliability between the PI and the ST microcontroller, creation of a P controller for the stepper motors, utilization of a more accurate sensor for the steering angle, data communication between the PI and a data logging laptop, communication between the PI and the GPS, and programming the SICK laser. However, much of the code is written as shown in, 0 and 0. Section 0 is the 'h files' for all the code used to program the robot while section 0 is the corresponding 'c files'. The 'main' file can be found in, 0. This is the starting point for the code. The rest of the code branches out from that.

It was desired to simulate the system is a manner in which it was to be utilized, such as paths that may be common to mowing-lawn or moving snow. This means paths that are both grid like and with planned maneuvers throughout to represent obstacles from landscaping. Figure 5.1 shows a proposed trajectory. This model doesn't have any obstacles built into it, but it does have turns that become tighter and tighter throughout the length of the path. This is meant to simulate the systems reaction to obstacles. The system may need to evade obstacles quickly which means a tight maneuver might need to be performed. The ability to track the path accurately in these circumstance is critical to mission performance.

Figure 5.1 could also be used for velocity testing. Once the speed limitations from the turning tests had been determined, tests that focus on the relationship between velocity and straight-line path tracking could be performed. The idea behind this is that the nonlinear model should run at a certain ratio of the velocity. However, it may be that hardware limitations prevent that ratio from being reached. This may have an effect on path tracking.

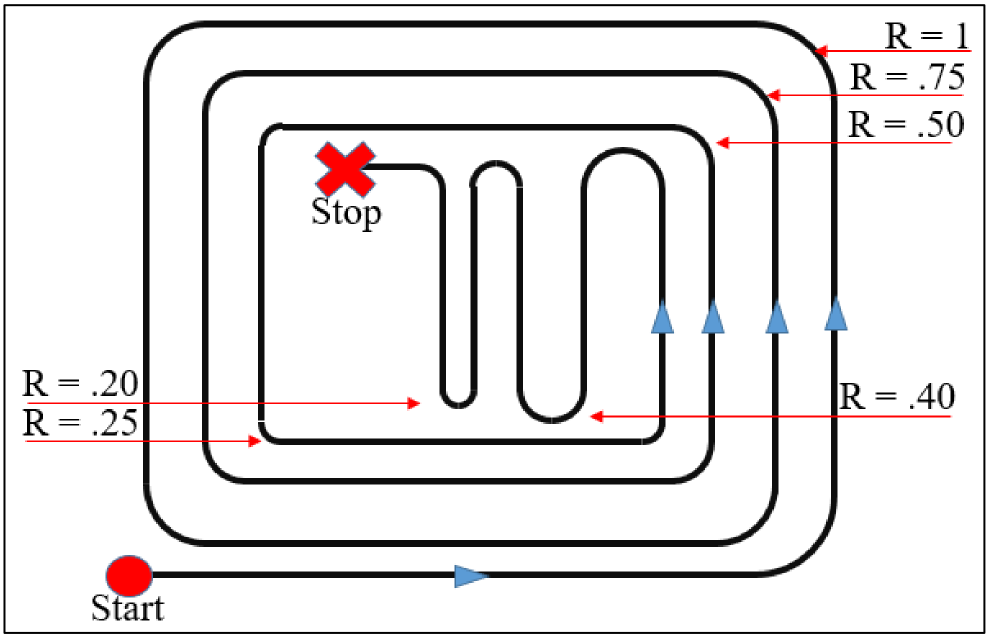

Figure 5.1: A proposed trajectory to test the dynamic model of the 4WD4WS system. The decreasing radius of the curves is meant to determine the path tracking accuracy under conditions of higher acceleration.

Thus, it should be determined what effect velocity has on path tracking performance and what amount of error is reasonable. It could be that a large velocity increase, resulting in a iteration to velocity ratio that is smaller than desirable, may result in an insignificant decrease in path tracking performance.

Tables could be made that show the error against velocity such that standards could be created for different tasks. Some tasks, like mowing of a professional baseball stadium, may require pinpoint accuracy, and a lower overall velocity, while other applications may be able to get by with less precision.

6. CONCLUDING REMARKS

This thesis has investigated the kinematic and dynamic modelling for a four-wheel independent-drive, four-wheel independent-steer robotic vehicle for the use as a consumer reconfigurable robotic system. The 4WD4WS structure was chosen to be a reconfigurable system due to its ability to handle many unique environments, such as grass and snow, while also maximizing mission performance.

The developed dynamic model boasts very high fidelity when compared to current models. Kinematic relationships guarantee that the system will never drive into obstacles which further increases the models path-tracking potential. This model is designed for a robot of arbitrary width and length, to follow a path of arbitrary steering angles, in a vehicle with arbitrary mass. The presents a system of incredible flexibility which means it can be incorporated into a wide variety of environments, including automotive, industrial, and consumer. This is in contrast to many models who limit their steering angles, use a vehicle of minimal mass, use a fixed frame size, or drive at small velocities in order to negate the effects of Newton's second law. While these models work well for their limited application, they are not flexible.

It became quite clear during the derivation of the model that it was far more complex than initially thought. The original plan was to create the dynamic model and advanced controller in tandem with the experimental system. While the experimental system was successfully created, the dynamic model kept growing in complexity and scale. Thus, the new focus of the project became the completion of the system's equations of motion. As discussed earlier, the dynamics are incredibly flexible. This is because we didn't take shortcuts, as many do, in order to get a simpler system. As a result, the flexible system is like none before it. With these equations, a follow up thesis would be in a very good spot to create a control algorithm to test with the completed experimental system.

6.1. Future Work

It is recommended that the current equations of motion be utilized to create a fully comprehensive control system that incorporates advanced control theory in conjunction with the advanced experimental vehicle. This would further prove the models effectiveness which may help the 4WD4WS structure be used in more mobile robots. It is also recommended that the control algorithm incorporate dynamic path planning. This would provide a system that would not only stay on the path but also avoid obstacles.

7. REFERENCES

[1] “Unimate - The First Industrial Robot,” *Robotics Online*. [Online]. Available: https://www.robotics.org/joseph-engelberger/unimate.cfm. [Accessed: 24-Oct-2017].

[2] Patrick Muir and C. Neuman, “Kinematic modeling of Wheeled Mobile Robots.pdf.” Jun-1986.

[3] T. Stewart, “Bruce, the multi-function robot,” Master of Fine Arts, Rochester Institute of Technology, Rochester, NY, 2012.

[4] “Top 5 Most Advanced Robotic Lawn Mowers,” *Into Robotics*. [Online]. Available: https://www.intorobotics.com/top-5-most-advanced-robotics-lawn-mowers/ [Accessed: 20-Jul-2017].

[5] “Roomba Robot Vacuum | iRobot.” [Online]. Available: http://www.irobot.com/For-the-Home/Vacuuming/Roomba.aspx. [Accessed: 20-Jul-2017].

[6] C. Bartle, Roomba Long Exposure. 2009. [Online]. Available: https://www.flickr.com/photos/13963375@N00/3533146556/. [Accessed: 20-Jul-2017].

[7] N. B. Ignell, N. Rasmusson, and J. Matsson, “An overview of legged and wheeled robotic locomotion,” *Available Mälardalen Univ. Web Site Httpwww Idt Mdh Sekurserct3340ht12MINICONFERENCEFinalPapersi Rcse12 Sub Mission*, vol. 21, 2012.

[8] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, “Bigdog, the rough-terrain quadruped robot,” *IFAC Proc. Vol.*, vol. 41, no. 2, pp. 10822–10825, 2008.

[9] “LS3 | Boston Dynamics.” [Online]. Available: https://www.bostondynamics.com/ls3. [Accessed: 22-Oct-2017].

[10] L. Plaugic, “Boston Dynamics’ redesigned Atlas robot is 75 percent more futuristic,” *The Verge*, 20-Jan-2015. [Online]. Available: https://www.theverge.com/2015/1/20/7857651/atlas-robot-unplugged-darpa-robotics-challenge.

[11] K. Goris, “Autonomous mobile robot mechanical design,” *VrijeUniversiteitBrussel Eng. Degree Thesis Bruss. Belg.*, 2005.

[12] R. Siegwart, I. Nourbakhsh, and D. Scaramuzza, “Introduction to Autonomous Mobile Robots.pdf.” MIT Press, 2011.

[13] E. Prassler, A. Ritter, C. Schaeffer, and P. Fiorini, “A short history of cleaning robots,” *Auton. Robots*, vol. 9, no. 3, pp. 211–226, 2000.

[14] T. Takamori and K. Tsuchiya, *Robotics, Mechatronics and Manufacturing Systems*. Elsevier, 2012.

[15] “MobileRobots Pioneer 3-AT (P3AT) research robot platform.” [Online]. Available: http://www.mobilerobots.com/ResearchRobots/P3AT.aspx. [Accessed: 23-Jul-2017].

[16] J. A. Batlle and A. Barjau, “Holonomy in mobile robots,” *Robot. Auton. Syst.*, vol. 57, no. 4, pp. 433–440, Apr. 2009.

[17] “Equipment - AG Technische Informatik - Universität Bielefeld.” [Online]. Available: https://www.ti.uni-bielefeld.de/html/research/equipment.html. [Accessed: 23-Jul-2017].

[18] J. Yi, D. Song, J. Zhang, and Z. Goodwin, “Adaptive Trajectory Tracking Control of Skid-Steered Mobile Robots,” in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 2605–2610.

[19] X. Wu, M. Xu, and L. Wang, “Differential speed steering control for four-wheel independent driving electric vehicle,” in *Industrial Electronics (ISIE), 2013 IEEE International Symposium on*, 2013, pp. 1–6.

[20] “Seekur Jr | Mobile Robot Platform for Outdoor Applications.” [Online]. Available: http://www.mobilerobots.com/ResearchRobots/SeekurJr.aspx. [Accessed: 23-Jul-2017].

[21] J. Morales, J. L. Martínez, A. Mandow, A. Pequeño-Boter, and A. García-Cerezo, “Simplified power consumption modeling and identification for wheeled skid-steer robotic vehicles on hard horizontal ground,” in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 4769–4774.

[22] H. Sayyaadi, H. Kouhi, and H. Salarieh, “Control of car-like (wheeled) multi robots for following and hunting a moving target,” *Sci. Iran.*, vol. 18, no. 4, pp. 950–965, Aug. 2011.

[23] “Wheeled Platform | AMBOT | American Robot Company.” [Online]. Available: http://www.ambot.com/ip-wheel.shtml. [Accessed: 23-Jul-2017].

[24] M. Ye, Q. Wang, and S. Jiao, “Robust H 2 / H ∞ Control for the Electrohydraulic Steering System of a Four-Wheel Vehicle,” *Math. Probl. Eng.*, vol. 2014, pp. 1–12, 2014.

[25] G. Reina, “Cross-Coupled Control for All-Terrain Rovers,” *Sensors*, vol. 13, no. 1, pp. 785–800, Jan. 2013.

[26] D. S. Choudhari, “Four Wheel Steering System For Future,” *Int. J. Mech. Eng. Robot. Res.*, vol. 3, no. 4, p. 383, 2014.

[27] N. Lashgarian Azad, “Dynamic modelling and stability controller development for articulated steer vehicles.,” Library and Archives Canada = Bibliothèque et Archives Canada, Ottawa, 2009.

[28] E. Haverluk, K. Hutchens, M. Muske, and V. O'Gara, "462 Final Paper Autonomous Snow Plow: ME 462 Final Report.pdf." NDSU Mechanical Engineering, 2014.

[29] D. Fox, W. Burgard, and S. Thrun, "Controlling synchro-drive robots with the dynamic window approach to collision avoidance," in *Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, 1996, vol. 3, pp. 1280–1287.

[30] T. Bräunl, *Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems*. Springer Science & Business Media, 2013.

[31] O. Diegel, A. Badve, G. Bright, J. Potgieter, and T. Sylvester, "Improved Mecanum Wheel Design for Omni-Directional Robots.pdf." Australian Conference on Robotics and Automation, Nov-2002.

[32] C. Sprunk, B. Lau, P. Pfaff, and W. Burgard, "An accurate and efficient navigation system for omnidirectional robots in industrial environments," *Auton. Robots*, vol. 41, no. 2, pp. 473–493, Feb. 2017.

[33] G. Indiveri, J. Paulus, and P. G. Plöger, "Motion Control of Swedish Wheeled Mobile Robots in the Presence of Actuator Saturation," in *RoboCup 2006: Robot Soccer World Cup X*, vol. 4434, G. Lakemeyer, E. Sklar, D. G. Sorrenti, and T. Takahashi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 35–46.

[34] M. Makatchev, S. K. Tso, S. Y. T. Lang, and J. J. McPhee, "Cross-coupling control for slippage minimization of a four-wheel-steering mobile robot," in *INTERNATIONAL SYMPOSIUM ON ROBOTICS*, 2000, vol. 31, pp. 42–47.

[35] V. Arvind, "Optimizing the turning radius of a vehicle using symmetric four wheel steering system," Dec-2013. [Online]. Available:

https://www.ijser.org/paper/Optimizing-the-turning-radius-of-a-vehicle-using-symmetric.html. [Accessed: 23-Jul-2017].

[36] R. Oftadeh, M. M. Aref, R. Ghabcheloo, and J. Mattila, "Bounded-velocity motion control of four wheel steered mobile robots," in *Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME International Conference on*, 2013, pp. 255–260.

[37] J. R. Nistler, "Optimum Path Tracking of an Independently Steered Four Wheeled Mobile Robot." North Dakota State University, Mar-2012.

[38] "Seekur | Outdoor Programmable Surveillance & Security Robot." [Online]. Available: http://www.mobilerobots.com/ResearchRobots/Seekur.aspx. [Accessed: 23-Jul-2017].

[39] R. Wang, C. Hu, Z. Wang, F. Yan, and N. Chen, "Integrated optimal dynamics control of 4WD4WS electric ground vehicle with tire-road frictional coefficient estimation," *Mech. Syst. Signal Process.*, vol. 60–61, pp. 727–741, Aug. 2015.

[40] C.-J. Lin, S.-M. Hsiao, Y.-H. Wang, C.-H. Yeh, C.-F. Huang, and T.-H. S. Li, "Design and implementation of a 4WS4WD mobile robot and its control applications," in *System Science and Engineering (ICSSE), 2013 International Conference on*, 2013, pp. 235–240.

[41] M. F. Selekwa and J. R. Nistler, "Path tracking control of four wheel independently steered ground robotic vehicles," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, 2011, pp. 6355–6360.

[42] R. Marino and F. Cinili, "Input–Output Decoupling Control by Measurement Feedback in Four-Wheel-Steering Vehicles," *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 5, pp. 1163–1172, Sep. 2009.

[43] A. Alfi and M. Farrokhi, “Hybrid state-feedback sliding-mode controller using fuzzy logic for four-wheel-steering vehicles,” *Veh. Syst. Dyn.*, vol. 47, no. 3, pp. 265–284, Mar. 2009.

[44] R. Marino, S. Scalzi, and F. Cinili, “Nonlinear PI front and rear steering control in four wheel steering vehicles,” *Veh. Syst. Dyn.*, vol. 45, no. 12, pp. 1149–1168, Dec. 2007.

[45] C. Chen, Y. Jia, J. Du, and F. Yu, “Lane keeping control for autonomous 4WS4WD vehicles subject to wheel slip constraint,” in *American Control Conference (ACC), 2012*, 2012, pp. 6515–6520.

[46] Chengliang Liu, Mingjun Wang, and Jun Zhou, “Coordinating control for an agricultural vehicle with individual wheel speeds and steering angles [Applications of Control],” *IEEE Control Syst. Mag.*, vol. 28, no. 5, pp. 21–24, Oct. 2008.

[47] P. Dai and J. Katupitiya, “Path planning and force control of a 4wd4ws vehicle,” in *Proceedings of Australasian Conference on Robotics and Automation*, 2014.

[48] L. DanYong and S. YongDuan, “Adaptive fault-tolerant tracking control of 4WS4WD road vehicles: A fully model-independent solution,” in *Control Conference (CCC), 2012 31st Chinese*, 2012, pp. 485–492.

[49] F. Fahimi, “Full drive-by-wire dynamic control for four-wheel-steer all-wheel-drive vehicles,” *Veh. Syst. Dyn.*, vol. 51, no. 3, pp. 360–376, Mar. 2013.

[50] T. D. Fields, *The development and optimization of a teleoperated four-wheel drive/four-wheel steer vehicle*. University of Nevada, Reno, 2009.

[51] E. P. Godoy, G. T. Tangerino, R. A. Tabile, R. Y. Inamasu, and A. J. V. Porto, “Networked Control System for the Guidance of a Four-Wheel Steering Agricultural Robotic Platform,” *J. Control Sci. Eng.*, vol. 2012, pp. 1–10, 2012.

[52] R. Grepl, J. Vejlupek, V. Lambersky, M. Jasansky, F. Vadlejch, and P. Coupek, “Development of 4WS/4WD Experimental Vehicle: platform for research and education in mechatronics,” in *Mechatronics (ICM), 2011 IEEE International Conference on*, 2011, pp. 893–898.

[53] Hao Yang, V. Cocquempot, and Bin Jiang, “Optimal Fault-Tolerant Path-Tracking Control for 4WS4WD Electric Vehicles,” *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 1, pp. 237–243, Mar. 2010.

[54] T. Hiraoka, O. Nishihara, and H. Kumamoto, “Automatic path-tracking controller of a four-wheel steering vehicle,” *Veh. Syst. Dyn.*, vol. 47, no. 10, pp. 1205–1227, Oct. 2009.

[55] S. Horiuchi, “Improvement of vehicle handling by nonlinear integrated control of four wheel steering and four wheel torque,” *JSAE Rev.*, vol. 20, no. 4, pp. 459–464, Oct. 1999.

[56] H. Karki, T. Yamashita, and K. Ichiryu, “Development of Field Robot,” in *Proceedings of the JFPS International Symposium on Fluid Power*, 2005, vol. 2005, pp. 301–304.

[57] R. Leenen, J. J. Ploeg, H. H. Nijmeijer, L. Moreau, and F. Veldpaus, “Motion control design for a 4ws and 4wd overactuated vehicle,” Master Thesis, Department Mechanical Engineering Dynamics and Control Technology Group, Eindhoven University of Technology, Eindhoven, 2004.

[58] P. M. Leucht, “Active Four-Wheel-Steering Design for an Advanced Vehicle,” in *American Control Conference, 1988*, 1988, pp. 2379–2384.

[59] M. Li and J. Yingmin, “Decoupling control in velocity varying four wheel steering vehicles with H∞ performance by longitudinal velocity and yaw rate feedback.pdf.” International Journal of Vehicle Mechanics and Mobility, 11-Sep-2014.

[60] E. Lucet, R. Lenain, and C. Grand, “Dynamic path tracking control of a vehicle on slippery terrain,” *Control Eng. Pract.*, vol. 42, pp. 60–73, Sep. 2015.

[61] M. Makatchev, J. J. McPhee, S. K. Tso, and S. Y. Lang, “System design modeling and control of a four-wheel-steering mobile robot,” in *Proc. 19th Chinese Control Conference*, 2000, pp. 759–763.

[62] O. Mokhiamar and M. Abe, “Simultaneous Optimal Distribution of Lateral and Longitudinal Tire Forces for the Model Following Control,” *J. Dyn. Syst. Meas. Control*, vol. 126, no. 4, p. 753, 2004.

[63] S.-T. Peng, “On One Approach to Constraining the Combined Wheel Slip in the Autonomous Control of a 4WS4WD Vehicle,” *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 1, pp. 168–175, Jan. 2007.

[64] R. Potluri and A. K. Singh, “Path-Tracking Control of an Autonomous 4WS4WD Electric Vehicle Using Its Natural Feedback Loops,” *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 5, pp. 2053–2062, Sep. 2015.

[65] A. Stotsky and X. Hu, “Adaptive/variable structure control of car-like mobile robot in four wheel dynamical model framework,” in *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, 1998, vol. 3, pp. 3111–3116.

[66] Tin Lun Lam, Huihuan Qian, and Yangsheng Xu, “Omnidirectional Steering Interface and Control for a Four-Wheel Independent Steering Vehicle,” *IEEEASME Trans. Mechatron.*, vol. 15, no. 3, pp. 329–338, Jun. 2010.

[67] T. Weiskircher and S. Müller, “Control performance of a road vehicle with four independent single-wheel electric motors and steer-by-wire system,” *Veh. Syst. Dyn.*, vol. 50, no. sup1, pp. 53–69, Jan. 2012.

[68] H. Xu, K. Xue, P. Wang, B. S. Marie, R. Wei, and B. Jin, “Maneuver control and kinematical energy analysis of a robot based on instantaneous Center of Rotation,” in *E-Learning in Industrial Electronics, 2006 1ST IEEE International Conference on*, 2006, pp. 101–106.

[69] Y. Yavin, “modeling the motion of a car with four steerable wheels.pdf.” Elsevier Ltd, May-2003.

[70] J. Ploeg, J. Vissers, and H. Nijmeijer, “Control Design for an Overactuated Wheeled Mobile Robot.pdf.” TNO Automotive, 2006.

[71] M. K. Yalcin, S. M. Yesiloglu, M. Dal, and H. Temeltas, “Maneuvering strategies for four-wheel drive, four-wheel steer mobile robots using curvatures based on weingarten-maps,” in *IEEE Industrial Electronics, IECon 2006-32nd Annual Conference on*, 2006, pp. 4148–4152.

[72] A. S. S. Committee, *The Corporate Report: A Discussion Paper Published for Comment*. Accounting Standards Steering Committee of the Institute of Chartered Accountants in England and Wales, 1975.

[73] Y. K. Tham, H. Wang, and E. K. Teoh, “Multi-sensor fusion for steerable four-wheeled industrial vehicles,” *Control Eng. Pract.*, vol. 7, no. 10, pp. 1233–1248, 1999.

[74] M. F. Selekwa, “Lecture 20: The Global Positioning System and Compasses.pdf.” North Dakota State University, 15-Apr-2015.

[75] “Circle, Cylinder, Sphere.” [Online]. Available: http://paulbourke.net/geometry/circlesphere/. [Accessed: 29-Aug-2017].

APPENDIX

A.1. MATLAB Simulation Code

A.1.1. Equations_Non_Symbolic.m

```
clear
clc
%% Defintion of Constants and Variables

Mb =   50; %kg
Ib =  5.5; %kg*m^2
Mi =  3.5; %kg
Ii = .025; %kg*m^2
rw = .085; %m
Is = .009; %kg*m^2
W = .75;  % meter;
L =   1;  % meter;

%    Xf   Yf   Xr   Yr   df   dr
q = [1; 1; 1; 1; 1; 1];

qd = [1; 1; 1; 1; 1; 1];

qdd = [1; 1; 1; 1; 1; 1];
%% Positions

% Initial Body Position
XBo = 0;
YB0 = 0;

% Initial Wheel Positions
X1o = L/2;  %2.44
X2o = L/2;  %2.44
X3o = -L/2; %2.44
X4o = -L/2; %2.44

Xo = zeros(4,1);
Xo(1,1) = X1o;
Xo(2,1) = X2o;
Xo(3,1) = X3o;
Xo(4,1) = X4o;

Y1o = W/2;  %2.45   %Note, the aspect ratio is not used
Y2o = -W/2; %2.45
Y3o = -W/2; %2.45
Y4o = W/2;  %2.45

Yo = zeros(4,1);
Yo(1,1) = Y1o;
Yo(2,1) = Y2o;
```

```
Yo(3,1) = Y3o;
Yo(4,1) = Y4o;

% Body Position
XB = .5*(q(1)-q(3)); %2.31
YB = .5*(q(2)-q(4)); %2.31

% Wheel-Body Positions
X1 = .5*(q(1)+q(3))+(1/L)*((q(1)-q(3))*X1o-(q(2)-q(4))*Y1o); %2.42
X2 = .5*(q(1)+q(3))+(1/L)*((q(1)-q(3))*X2o-(q(2)-q(4))*Y2o);
X3 = .5*(q(1)+q(3))+(1/L)*((q(1)-q(3))*X3o-(q(2)-q(4))*Y3o);
X4 = .5*(q(1)+q(3))+(1/L)*((q(1)-q(3))*X4o-(q(2)-q(4))*Y4o);

Y1 = .5*(q(2)+q(4))+(1/L)*((q(2)-q(4))*X1o+(q(1)-q(3))*Y1o); %2.42
Y2 = .5*(q(2)+q(4))+(1/L)*((q(2)-q(4))*X2o+(q(1)-q(3))*Y2o);
Y3 = .5*(q(2)+q(4))+(1/L)*((q(2)-q(4))*X3o+(q(1)-q(3))*Y3o);
Y4 = .5*(q(2)+q(4))+(1/L)*((q(2)-q(4))*X4o+(q(1)-q(3))*Y4o);

% ICR Position
X0 = -(q(4)*cos(q(5))*cos(q(6))-
q(2)*cos(q(5))*cos(q(6))+q(1)*cos(q(5))*sin(q(6))+q(3)*cos(q(6))*sin(q
(5)))/sin(q(5)+q(6));   %2.36
Y0 = -(q(1)*cos(q(5))*cos(q(6))-
q(3)*cos(q(5))*cos(q(6))+q(2)*cos(q(5))*sin(q(6))+q(4)*cos(q(6))*sin(q
(5)))/sin(q(5)+q(6));   %2.39

%% pi definitions

pf = L*cos(q(6))/(sin(q(5)+q(6)));
pr = L*cos(q(5))/(sin(q(5)+q(6)));

% (pisq represents the squared distance from the wheel-body, i, to the
ICR,
p1sq = (X1-X0)^2+(Y1-Y0)^2;
p2sq = (X2-X0)^2+(Y2-Y0)^2;
p3sq = (X3-X0)^2+(Y3-Y0)^2;
p4sq = (X4-X0)^2+(Y4-Y0)^2;

%pi represents the distance from the wheel-body, i, to the ICR,
p1 =
L*cos(q(6))/sin(q(5)+q(6))*(1+W*cos(q(5))*sin(q(5)+q(6))/(L*cos(q(6)))
+(W*sin(q(5)+q(6))/(2*L*cos(q(6))))^2)^.5;   %positive for left
p2 = L*cos(q(6))/sin(q(5)+q(6))*(1-
W*cos(q(5))*sin(q(5)+q(6))/(L*cos(q(6)))+(W*sin(q(5)+q(6))/(2*L*cos(q(
6))))^2)^.5;   %negative for right
p3 = L*cos(q(5))/sin(q(5)+q(6))*(1-
W*cos(q(6))*sin(q(5)+q(6))/(L*cos(q(5)))+(W*sin(q(5)+q(6))/(2*L*cos(q(
5))))^2)^.5;   %negative for right
p4 =
L*cos(q(5))/sin(q(5)+q(6))*(1+W*cos(q(6))*sin(q(5)+q(6))/(L*cos(q(5)))
+(W*sin(q(5)+q(6))/(2*L*cos(q(5))))^2)^.5;   %positive for left
```

```
PR = zeros(4,1);     %These values for ro are used in the Generalized
Force and EOM equations
PR(1) = p1;
PR(2) = p2;
PR(3) = p3;
PR(4) = p4;

%% J
%The Jxi terms from Equation (2.47) and (2.48)
J = zeros(2,4);

%Jxi
J(1,1) = q(1)-((q(1)-q(3))*sin(q(5))+(q(2)-
q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))-
(.5*(q(1)+q(3))+(1/L)*((q(1)-q(3))*X1o-(q(2)-q(4))*Y1o));
J(1,2) = q(1)-((q(1)-q(3))*sin(q(5))+(q(2)-
q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))-
(.5*(q(1)+q(3))+(1/L)*((q(1)-q(3))*X2o-(q(2)-q(4))*Y2o));
J(1,3) = q(1)-((q(1)-q(3))*sin(q(5))+(q(2)-
q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))-
(.5*(q(1)+q(3))+(1/L)*((q(1)-q(3))*X3o-(q(2)-q(4))*Y3o));
J(1,4) = q(1)-((q(1)-q(3))*sin(q(5))+(q(2)-
q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))-
(.5*(q(1)+q(3))+(1/L)*((q(1)-q(3))*X4o-(q(2)-q(4))*Y4o));

%Jyi
J(2,1) = q(2)-((q(1)-q(3))*sin(q(5))-(q(2)-
q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))-
(.5*(q(2)+q(4))+(1/L)*((q(2)-q(4))*X1o+(q(1)-q(3))*Y1o));
J(2,2) = q(2)-((q(1)-q(3))*sin(q(5))-(q(2)-
q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))-
(.5*(q(2)+q(4))+(1/L)*((q(2)-q(4))*X2o+(q(1)-q(3))*Y2o));
J(2,3) = q(2)-((q(1)-q(3))*sin(q(5))-(q(2)-
q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))-
(.5*(q(2)+q(4))+(1/L)*((q(2)-q(4))*X3o+(q(1)-q(3))*Y3o));
J(2,4) = q(2)-((q(1)-q(3))*sin(q(5))-(q(2)-
q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))-
(.5*(q(2)+q(4))+(1/L)*((q(2)-q(4))*X4o+(q(1)-q(3))*Y4o));

%% A
% The Aik terms from Equation 2.60, subscripts are wheel (i), then
term (k)
A = zeros(4,6);

A(1,1) = (1/2 + X1o/L);
A(2,1) = (1/2 + X2o/L);
A(3,1) = (1/2 + X3o/L);
A(4,1) = (1/2 + X4o/L);

A(1,2) = -(Y1o/L);
A(2,2) = -(Y2o/L);
A(3,2) = -(Y3o/L);
```

```
A(4,2) = -(Y4o/L);

A(1,3) = (1/2 - X1o/L);
A(2,3) = (1/2 - X2o/L);
A(3,3) = (1/2 - X3o/L);
A(4,3) = (1/2 - X4o/L);

A(1,4) = (Y1o/L);
A(2,4) = (Y2o/L);
A(3,4) = (Y3o/L);
A(4,4) = (Y4o/L);

A(1,5) = 0;
A(2,5) = 0;
A(3,5) = 0;
A(4,5) = 0;

A(1,6) = 0;
A(2,6) = 0;
A(3,6) = 0;
A(4,6) = 0;
%% B
% The Bik terms from Equation 2.62, subscripts are wheel (i), then
term (k)
B = zeros(4,6);

B(1,1) = (Y1o/L);
B(2,1) = (Y2o/L);
B(3,1) = (Y3o/L);
B(4,1) = (Y4o/L);

B(1,2) = (1/2 + X1o/L);
B(2,2) = (1/2 + X2o/L);
B(3,2) = (1/2 + X3o/L);
B(4,2) = (1/2 + X4o/L);

B(1,3) = -(Y1o/L);
B(2,3) = -(Y2o/L);
B(3,3) = -(Y3o/L);
B(4,3) = -(Y4o/L);

B(1,4) = (1/2 - X1o/L);
B(2,4) = (1/2 - X2o/L);
B(3,4) = (1/2 - X3o/L);
B(4,4) = (1/2 - X4o/L);

B(1,5) = 0;
B(2,5) = 0;
B(3,5) = 0;
B(4,5) = 0;

B(1,6) = 0;
```

```
B(2,6) = 0;
B(3,6) = 0;
B(4,6) = 0;

%% H
% The Hxi and Hyi terms from Equation 2.66 and 2.68
H = zeros(2,6);
H(1,1) = (1-sin(q(5))*cos(q(6))*csc(q(5)+q(6)));
H(1,2) = -cos(q(5))*cos(q(6))*csc(q(5)+q(6));
H(1,3) =  sin(q(5))*cos(q(6))*csc(q(5)+q(6));
H(1,4) =  cos(q(5))*cos(q(6))*csc(q(5)+q(6));
H(1,5) = ((((q(2)-q(4))*sin(q(5))-(q(1)-
q(3))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6)))-((q(1)-
q(3))*sin(q(5))+(q(2)-
q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6)));
H(1,6) = (-(q(1)-q(3))*sin(q(5))+(q(2)-
q(4))*cos(q(5)))*csc(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6)));

H(2,1) =  -cos(q(5))*cos(q(6))*csc(q(5)+q(6));
H(2,2) =(1+cos(q(5))*cos(q(6))*csc(q(5)+q(6)));
H(2,3) =  sin(q(5))*cos(q(6))*csc(q(5)+q(6));
H(2,4) = -cos(q(5))*cos(q(6))*csc(q(5)+q(6));
H(2,5) = ((((q(2)-q(4))*sin(q(5))-(q(1)-
q(3))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6)))-((q(1)-q(3))*sin(q(5))-
(q(2)-q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6)));
H(2,6) = -((q(1)-q(3))*sin(q(5))-(q(2)-
q(4))*cos(q(5)))*csc(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6)));

% Derivatives of the H1 and H2 matrices.
Hx = zeros(6,6);
Hx(1,1) = 0;
Hx(1,2) = 0;
Hx(1,3) = 0;
Hx(1,4) = 0;
Hx(1,5) = (-
cos(q(5))*cos(q(6))+sin(q(5))*cos(q(6))*cot(q(5)+q(6)))*csc(q(5)+q(6))
;
Hx(1,6) =
(sin(q(5))*sin(q(6))+sin(q(5))*cos(q(6))*cot(q(5)+q(6)))*csc(q(5)+q(6)
);

Hx(2,1) = 0;
Hx(2,2) = 0;
Hx(2,3) = 0;
Hx(2,4) = 0;
Hx(2,5) =
(sin(q(5))*cos(q(6))+cos(q(5))*cos(q(6))*cot(q(5)+q(6)))*csc(q(5)+q(6)
);
Hx(2,6) =
(cos(q(5))*sin(q(6))+cos(q(5))*cos(q(6))*cot(q(5)+q(6)))*csc(q(5)+q(6)
);
```

```
Hx(3,1) = 0;
Hx(3,2) = 0;
Hx(3,3) = 0;
Hx(3,4) = 0;
Hx(3,5) = (cos(q(5))*cos(q(6))-
sin(q(5))*cos(q(6))*cot(q(5)+q(6)))*csc(q(5)+q(6));
Hx(3,6) = (-sin(q(5))*sin(q(6))-
sin(q(5))*cos(q(6))*cot(q(5)+q(6)))*csc(q(5)+q(6));

Hx(4,1) = 0;
Hx(4,2) = 0;
Hx(4,3) = 0;
Hx(4,4) = 0;
Hx(4,5) = (-sin(q(5))*cos(q(6))-
cos(q(5))*cos(q(6))*cot(q(5)+q(6)))*csc(q(5)+q(6));
Hx(4,6) = (-cos(q(5))*sin(q(6))-
cos(q(5))*cos(q(6))*cot(q(5)+q(6)))*csc(q(5)+q(6));

Hx(5,1) = (-cos(q(5))*cos(q(6))*csc(q(5)+q(6))-
sin(q(5))*cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6)));%
Hx(5,2) = (sin(q(5))*cos(q(6))*csc(q(5)+q(6))-
cos(q(5))*cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6)));%
Hx(5,3) =
(cos(q(5))*cos(q(6))*csc(q(5)+q(6))+sin(q(5))*cos(q(6))*csc(q(5)+q(6))
*cot(q(5)+q(6)));%
Hx(5,4) = (-
sin(q(5))*cos(q(6))*csc(q(5)+q(6))+cos(q(5))*cos(q(6))*csc(q(5)+q(6))*
cot(q(5)+q(6)));%
Hx(5,5) = ((q(2)-q(4))*cos(q(5))+(q(1)-
q(3))*sin(q(5)))*cos(q(6))*csc(q(5)+q(6))-((q(2)-q(4))*sin(q(5))-
(q(1)-q(3))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6))...
          -((q(1)-q(3))*cos(q(5))-(q(2)-
q(4))*sin(q(5)))*cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6))...
          +((q(1)-q(3))*sin(q(5))+(q(2)-
q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6))^2+((q(1)-
q(3))*sin(q(5))+(q(2)-q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))^3; %
Hx(5,6) = ((q(2)-q(4))*sin(q(5))-(q(1)-q(3))*cos(q(5)))*(-
sin(q(6))*csc(q(5)+q(6))-cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6)))...
          -((q(1)-q(3))*sin(q(5))+(q(2)-q(4))*cos(q(5)))*(-
sin(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6))-
cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6))^2-cos(q(6))*csc(q(5)+q(6))^3);
%

Hx(6,1) = (-
sin(q(5))*csc(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6))));%
Hx(6,2) = (-
cos(q(5))*csc(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6))));%
Hx(6,3) =
(sin(q(5))*csc(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6))));%
Hx(6,4) =
(cos(q(5))*csc(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6)))); %
```

```
Hx(6,5) = -((q(1)-q(3))*cos(q(5))-(q(2)-
q(4))*sin(q(5)))*csc(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6)))..
.
          -((q(1)-q(3))*sin(q(5))+(q(2)-q(4))*cos(q(5)))*(-
csc(q(5)+q(6))*cot(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6)))-
cos(q(6))*csc(q(5)+q(6))^3); %
Hx(6,6) = -((q(1)-q(3))*sin(q(5))+(q(2)-q(4))*cos(q(5)))*(-
csc(q(5)+q(6))*cot(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6)))+csc
(q(5)+q(6))*(cos(q(6))-sin(q(6))*cot(q(5)+q(6))-
cos(q(6))*csc(q(5)+q(6))^2)); %

Hy = zeros(6,6);
Hy(1,1) = 0;
Hy(1,2) = 0;
Hy(1,3) = 0;
Hy(1,4) = 0;
Hy(1,5) =
(sin(q(5))*cos(q(6))+cos(q(5))*cos(q(6))*cot(q(5)+q(6)))*csc(q(5)+q(6)
);
Hy(1,6) =
(cos(q(5))*sin(q(6))+cos(q(5))*cos(q(6))*cot(q(5)+q(6)))*csc(q(5)+q(6)
);

Hy(2,1) = 0;
Hy(2,2) = 0;
Hy(2,3) = 0;
Hy(2,4) = 0;
Hy(2,5) = (-sin(q(5))*cos(q(6))-
cos(q(5))*cos(q(6))*cot(q(5)+q(6)))*csc(q(5)+q(6));
Hy(2,6) = (cos(q(5))*sin(q(6))-
cos(q(5))*cos(q(6))*cot(q(5)+q(6)))*csc(q(5)+q(6));

Hy(3,1) = 0;
Hy(3,2) = 0;
Hy(3,3) = 0;
Hy(3,4) = 0;
Hy(3,5) = (cos(q(5))*cos(q(6))-
sin(q(5))*cos(q(6))*cot(q(5)+q(6)))*csc(q(5)+q(6));
Hy(3,6) = (-sin(q(5))*sin(q(6))-
sin(q(5))*cos(q(6))*cot(q(5)+q(6)))*csc(q(5)+q(6));

Hy(4,1) = 0;
Hy(4,2) = 0;
Hy(4,3) = 0;
Hy(4,4) = 0;
Hy(4,5) =
(sin(q(5))*cos(q(6))+cos(q(5))*cos(q(6))*cot(q(5)+q(6)))*csc(q(5)+q(6)
);
Hy(4,6) = (cos(q(5))*sin(q(6))-
cos(q(5))*cos(q(6))*cot(q(5)+q(6)))*csc(q(5)+q(6));
```

```
Hy(5,1) = (-cos(q(5))*cos(q(6))*csc(q(5)+q(6))-
sin(q(5))*cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6)));
Hy(5,2) =
(sin(q(5))*cos(q(6))*csc(q(5)+q(6))+cos(q(5))*cos(q(6))*csc(q(5)+q(6))
*cot(q(5)+q(6)));
Hy(5,3) =
(cos(q(5))*cos(q(6))*csc(q(5)+q(6))+sin(q(5))*cos(q(6))*csc(q(5)+q(6))
*cot(q(5)+q(6)));
Hy(5,4) = (-sin(q(5))*cos(q(6))*csc(q(5)+q(6))-
cos(q(5))*cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6)));
Hy(5,5) = ((q(2)-q(4))*cos(q(5))+(q(1)-
q(3))*sin(q(5)))*cos(q(6))*csc(q(5)+q(6))-((q(2)-q(4))*sin(q(5))-
(q(1)-q(3))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6))...
          -((q(1)-q(3))*cos(q(5))+(q(2)-
q(4))*sin(q(5)))*cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6))...
          +((q(1)-q(3))*sin(q(5))-(q(2)-
q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6))^2+((q(1)-
q(3))*sin(q(5))-(q(2)-q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))^3; %
Hy(5,6) =  ((q(2)-q(4))*sin(q(5))-(q(1)-q(3))*cos(q(5)))*(-
sin(q(6))*csc(q(5)+q(6))-cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6)))...
          -((q(1)-q(3))*sin(q(5))+(q(2)-q(4))*cos(q(5)))*(-
sin(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6))-
cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6))^2-cos(q(6))*csc(q(5)+q(6))^3);
%


Hy(6,1) = (-
sin(q(5))*csc(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6))));%
Hy(6,2) =
(cos(q(5))*csc(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6))));%
Hy(6,3) =
(sin(q(5))*csc(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6))));%
Hy(6,4) = (-
cos(q(5))*csc(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6)))); %
Hy(6,5) = -((q(1)-q(3))*cos(q(5))+(q(2)-
q(4))*sin(q(5)))*csc(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6)))..
.
          -((q(1)-q(3))*sin(q(5))-(q(2)-q(4))*cos(q(5)))*(-
csc(q(5)+q(6))*cot(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6)))-
cos(q(6))*csc(q(5)+q(6))^3);%
Hy(6,6) = -((q(1)-q(3))*sin(q(5))-(q(2)-q(4))*cos(q(5)))*(-
csc(q(5)+q(6))*cot(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6)))+csc
(q(5)+q(6))*(cos(q(6))-sin(q(6))*cot(q(5)+q(6))-
cos(q(6))*csc(q(5)+q(6))^2)); %


%% S (Wheel Yaw-Rate)
% The Sik terms from Equation 2.71, subscripts are wheel (i), then
term (k)
S = zeros(4,6);
S(1,1) = (1/p1^2)*(J(2,1)*(A(1,1)-H(1,1))-J(1,1)*(B(1,1)-H(2,1)));
S(1,2) = (1/p1^2)*(J(2,1)*(A(1,2)-H(1,2))-J(1,1)*(B(1,2)-H(2,2)));
```

```
S(1,3) = (1/p1^2)*(J(2,1)*(A(1,3)-H(1,3))-J(1,1)*(B(1,3)-H(2,3)));
S(1,4) = (1/p1^2)*(J(2,1)*(A(1,4)-H(1,4))-J(1,1)*(B(1,4)-H(2,4)));
S(1,5) = (1/p1^2)*(J(2,1)*(A(1,5)-H(1,5))-J(1,1)*(B(1,5)-H(2,5)));
S(1,6) = (1/p1^2)*(J(2,1)*(A(1,6)-H(1,6))-J(1,1)*(B(1,6)-H(2,6)));

S(2,1) = (1/p2^2)*(J(2,2)*(A(2,1)-H(1,1))-J(1,2)*(B(2,1)-H(2,1)));
S(2,2) = (1/p2^2)*(J(2,2)*(A(2,2)-H(1,2))-J(1,2)*(B(2,2)-H(2,2)));
S(2,3) = (1/p2^2)*(J(2,2)*(A(2,3)-H(1,3))-J(1,2)*(B(2,3)-H(2,3)));
S(2,4) = (1/p2^2)*(J(2,2)*(A(2,4)-H(1,4))-J(1,2)*(B(2,4)-H(2,4)));
S(2,5) = (1/p2^2)*(J(2,2)*(A(2,5)-H(1,5))-J(1,2)*(B(2,5)-H(2,5)));
S(2,6) = (1/p2^2)*(J(2,2)*(A(2,6)-H(1,6))-J(1,2)*(B(2,6)-H(2,6)));

S(3,1) = (1/p3^2)*(J(2,3)*(A(3,1)-H(1,1))-J(1,3)*(B(3,1)-H(2,1)));
S(3,2) = (1/p3^2)*(J(2,3)*(A(3,2)-H(1,2))-J(1,3)*(B(3,2)-H(2,2)));
S(3,3) = (1/p3^2)*(J(2,3)*(A(3,3)-H(1,3))-J(1,3)*(B(3,3)-H(2,3)));
S(3,4) = (1/p3^2)*(J(2,3)*(A(3,4)-H(1,4))-J(1,3)*(B(3,4)-H(2,4)));
S(3,5) = (1/p3^2)*(J(2,3)*(A(3,5)-H(1,5))-J(1,3)*(B(3,5)-H(2,5)));
S(3,6) = (1/p3^2)*(J(2,3)*(A(3,6)-H(1,6))-J(1,3)*(B(3,6)-H(2,6)));

S(4,1) = (1/p4^2)*(J(2,4)*(A(4,1)-H(1,1))-J(1,4)*(B(4,1)-H(2,1)));
S(4,2) = (1/p4^2)*(J(2,4)*(A(4,2)-H(1,2))-J(1,4)*(B(4,2)-H(2,2)));
S(4,3) = (1/p4^2)*(J(2,4)*(A(4,3)-H(1,3))-J(1,4)*(B(4,3)-H(2,3)));
S(4,4) = (1/p4^2)*(J(2,4)*(A(4,4)-H(1,4))-J(1,4)*(B(4,4)-H(2,4)));
S(4,5) = (1/p4^2)*(J(2,4)*(A(4,5)-H(1,5))-J(1,4)*(B(4,5)-H(2,5)));
S(4,6) = (1/p4^2)*(J(2,4)*(A(4,6)-H(1,6))-J(1,4)*(B(4,6)-H(2,6)));

%% G (Body Velocities and Yaw-Rate)
% The Gik terms from Equations: 2.73, 2.75, 2.78, subscripts are wheel
(i), then term (k)
G = zeros(3,6);
G(1,1) = 1/2;
G(1,2) =   0;
G(1,3) = 1/2;
G(1,4) =   0;
G(1,5) =   0;
G(1,6) =   0;

G(2,1) =   0;
G(2,2) = 1/2;
G(2,3) =   0;
G(2,4) = 1/2;
G(2,5) =   0;
G(2,6) =   0;

G(3,1) = -(1/L^2)*(q(2)-q(4));
G(3,2) =  (1/L^2)*(q(1)-q(3));
G(3,3) =  (1/L^2)*(q(2)-q(4));
G(3,4) = -(1/L^2)*(q(1)-q(3));
G(3,5) =   0;
G(3,6) =   0;

%% K (last two constraints)
```

```
% Kfk is row 1 while Krk is row 2
K = zeros(2,6); %Eq 2.134
K(1,1) = (1/L)*((q(1)-q(3))*cos(q(5))-(q(2)-q(4))*sin(q(5)));
K(1,2) = (1/L)*((q(2)-q(4))*cos(q(5))+(q(1)-q(3))*sin(q(5)));
K(1,3) = 0;
K(1,4) = 0;
K(1,5) = 0;
K(1,6) = 0;

K(2,1) = (1/L)*((q(1)-q(3))*cos(q(6))+(q(2)-q(4))*sin(q(6)));
K(2,2) = (1/L)*((q(2)-q(4))*cos(q(6))-(q(1)-q(3))*sin(q(6)));
K(2,3) = 0;
K(2,4) = 0;
K(2,5) = 0;
K(2,6) = 0;

%% D
% Dfx is D(1,1), Dfy is D(1,2), Drx is D(2,1), Dry is D(2,2)
D = zeros(2,2);
D(1,1) =  ((q(1)-q(3))*sin(q(5))+(q(2)-
q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6));
D(1,2) =  ((q(1)-q(3))*sin(q(5))-(q(2)-
q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6));
D(2,1) =  ((q(1)-q(3))*cos(q(6))-(q(2)-
q(4))*sin(q(6)))*cos(q(5))*csc(q(5)+q(6));
D(2,2) = -((q(1)-q(3))*cos(q(6))+(q(2)-
q(4))*sin(q(6)))*cos(q(5))*csc(q(5)+q(6));

%% N
% Nxi = N(1,i),Nyi = N(2,i)
N = zeros(2,6);
N(1,1) =  sin(q(5))*cos(q(6))*csc(q(5)+q(6));
N(1,2) =  cos(q(5))*cos(q(6))*csc(q(5)+q(6));
N(1,3) = -sin(q(5))*cos(q(6))*csc(q(5)+q(6));
N(1,4) = -cos(q(5))*cos(q(6))*csc(q(5)+q(6));
N(1,5) = ((((q(1)-q(3))*cos(q(5))-(q(2)-
q(4))*sin(q(5)))*cos(q(6))*csc(q(5)+q(6)))-((q(1)-
q(3))*sin(q(5))+(q(2)-
q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6)));
N(1,6) = (-(q(1)-q(3))*sin(q(5))+(q(2)-
q(4))*cos(q(5)))*csc(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6)));

N(2,1) =  sin(q(5))*cos(q(6))*csc(q(5)+q(6));
N(2,2) = -cos(q(5))*cos(q(6))*csc(q(5)+q(6));
N(2,3) = -sin(q(5))*cos(q(6))*csc(q(5)+q(6));
N(2,4) =  cos(q(5))*cos(q(6))*csc(q(5)+q(6));
N(2,5) = ((((q(1)-q(3))*cos(q(5))+(q(2)-
q(4))*sin(q(5)))*cos(q(6))*csc(q(5)+q(6)))-((q(1)-
q(3))*sin(q(5))+(q(2)-
q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6)));
N(2,6) = (-(q(1)-q(3))*sin(q(5))-(q(2)-
q(4))*cos(q(5)))*csc(q(5)+q(6))*(sin(q(6))+cos(q(6))*cot(q(5)+q(6)));
```

```
%% U
% Uxi = U(1,i),Uyi = N(2,i)
U = zeros(2,6);
U(1,1) =  cos(q(5))*cos(q(6))*csc(q(5)+q(6));
U(1,2) = -cos(q(5))*sin(q(6))*csc(q(5)+q(6));
U(1,3) = -cos(q(5))*cos(q(6))*csc(q(5)+q(6));
U(1,4) =  cos(q(5))*sin(q(6))*csc(q(5)+q(6));
U(1,5) = -(((q(1)-q(3))*cos(q(6))-(q(2)-
q(4))*sin(q(6)))*csc(q(5)+q(6))*(sin(q(6))+cos(q(5))*cot(q(5)+q(6))));
U(1,6) = -(((q(1)-q(3))*sin(q(6))+(q(2)-
q(4))*cos(q(6)))*cos(q(5))*csc(q(5)+q(6))+((q(1)-q(3))*cos(q(6))-
(q(2)-q(4))*sin(q(6)))*cos(q(5))*csc(q(5)+q(6))*cot(q(5)+q(6)));

U(2,1) = -cos(q(5))*cos(q(6))*csc(q(5)+q(6));
U(2,2) = -cos(q(5))*sin(q(6))*csc(q(5)+q(6));
U(2,3) =  cos(q(5))*cos(q(6))*csc(q(5)+q(6));
U(2,4) =  cos(q(5))*sin(q(6))*csc(q(5)+q(6));
U(2,5) =  (((q(1)-q(3))*cos(q(6))+(q(2)-
q(4))*sin(q(6)))*csc(q(5)+q(6))*(sin(q(6))+cos(q(5))*cot(q(5)+q(6))));
U(2,6) =  (((q(1)-q(3))*sin(q(6))-(q(2)-
q(4))*cos(q(6)))*cos(q(5))*csc(q(5)+q(6))+((q(1)-
q(3))*cos(q(6))+(q(2)-
q(4))*sin(q(6)))*cos(q(5))*csc(q(5)+q(6))*cot(q(5)+q(6)));

%% Constraint Matrix C
C = zeros(9,6);

C(1,1) =  (q(1)-q(3));
C(1,2) =  (q(2)-q(4));
C(1,3) = -(q(1)-q(3));
C(1,4) = -(q(2)-q(4));
C(1,5) = 0;
C(1,6) = 0;

C(2,1) = (D(1,1)*N(1,1)+D(1,2)*N(2,1))*cos(q(5))^2-
(D(2,1)*U(1,1)+D(2,2)*U(2,1))*cos(q(6))^2;
C(2,2) = (D(1,1)*N(1,2)+D(1,2)*N(2,2))*cos(q(5))^2-
(D(2,1)*U(1,2)+D(2,2)*U(2,2))*cos(q(6))^2;
C(2,3) = (D(1,1)*N(1,3)+D(1,2)*N(2,3))*cos(q(5))^2-
(D(2,1)*U(1,3)+D(2,2)*U(2,3))*cos(q(6))^2;
C(2,4) = (D(1,1)*N(1,4)+D(1,2)*N(2,4))*cos(q(5))^2-
(D(2,1)*U(1,4)+D(2,2)*U(2,4))*cos(q(6))^2;
C(2,5) = (D(1,1)*N(1,5)+D(1,2)*N(2,5))*cos(q(5))^2-
(D(2,1)*U(1,5)+D(2,2)*U(2,5))*cos(q(6))^2-
(D(1,1)^2+D(1,2)^2)*cos(q(5))*sin(q(5));
C(2,6) = (D(1,1)*N(1,6)+D(1,2)*N(2,6))*cos(q(5))^2-
(D(2,1)*U(1,6)+D(2,2)*U(2,6))*cos(q(6))^2+(D(2,1)^2+D(2,2)^2)*cos(q(6)
)*sin(q(6));

C(3,1) = (D(2,1)*U(1,1)+D(2,2)*U(2,1))*sin(q(5)+q(6))^2;
C(3,2) = (D(2,1)*U(1,2)+D(2,2)*U(2,2))*sin(q(5)+q(6))^2;
```

```
C(3,3) = (D(2,1)*U(1,3)+D(2,2)*U(2,3))*sin(q(5)+q(6))^2;
C(3,4) = (D(2,1)*U(1,4)+D(2,2)*U(2,4))*sin(q(5)+q(6))^2;
C(3,5) =
(D(2,1)*U(1,5)+D(2,2)*U(2,5))*sin(q(5)+q(6))^2+(D(1,1)^2+D(1,2)^2)*cos
(q(5)+q(6))*sin(q(5)+q(6))+L^2*sin(q(5))*cos(q(5));
C(3,6) =
(D(2,1)*U(1,6)+D(2,2)*U(2,6))*sin(q(5)+q(6))^2+(D(1,1)^2+D(1,2)^2)*cos
(q(5)+q(6))*sin(q(5)+q(6));

C(4,1) = (1/p1^2)*(A(1,1)*J(1,1)-B(1,1)*J(2,1))-
(1/p2^2)*(A(2,1)*J(1,2)-B(2,1)*J(2,2));
C(4,2) = (1/p1^2)*(A(1,2)*J(1,1)-B(1,2)*J(2,1))-
(1/p2^2)*(A(2,2)*J(1,2)-B(2,2)*J(2,2));
C(4,3) = (1/p1^2)*(A(1,3)*J(1,1)-B(1,3)*J(2,1))-
(1/p2^2)*(A(2,3)*J(1,2)-B(2,3)*J(2,2));
C(4,4) = (1/p1^2)*(A(1,4)*J(1,1)-B(1,4)*J(2,1))-
(1/p2^2)*(A(2,4)*J(1,2)-B(2,4)*J(2,2));
C(4,5) = (1/p1^2)*(A(1,5)*J(1,1)-B(1,5)*J(2,1))-
(1/p2^2)*(A(2,5)*J(1,2)-B(2,5)*J(2,2));
C(4,6) = (1/p1^2)*(A(1,6)*J(1,1)-B(1,6)*J(2,1))-
(1/p2^2)*(A(2,6)*J(1,2)-B(2,6)*J(2,2));

C(5,1) = (1/p2^2)*(A(2,1)*J(1,2)-B(2,1)*J(2,2))-
(1/p3^2)*(A(3,1)*J(1,3)-B(3,1)*J(2,3));
C(5,2) = (1/p2^2)*(A(2,2)*J(1,2)-B(2,2)*J(2,2))-
(1/p3^2)*(A(3,2)*J(1,3)-B(3,2)*J(2,3));
C(5,3) = (1/p2^2)*(A(2,3)*J(1,2)-B(2,3)*J(2,2))-
(1/p3^2)*(A(3,3)*J(1,3)-B(3,3)*J(2,3));
C(5,4) = (1/p2^2)*(A(2,4)*J(1,2)-B(2,4)*J(2,2))-
(1/p3^2)*(A(3,4)*J(1,3)-B(3,4)*J(2,3));
C(5,5) = (1/p2^2)*(A(2,5)*J(1,2)-B(2,5)*J(2,2))-
(1/p3^2)*(A(3,5)*J(1,3)-B(3,5)*J(2,3));
C(5,6) = (1/p2^2)*(A(2,6)*J(1,2)-B(2,6)*J(2,2))-
(1/p3^2)*(A(3,6)*J(1,3)-B(3,6)*J(2,3));

C(6,1) = (1/p3^2)*(A(3,1)*J(1,3)-B(3,1)*J(2,3))-
(1/p4^2)*(A(4,1)*J(1,4)-B(4,1)*J(2,4));
C(6,2) = (1/p3^2)*(A(3,2)*J(1,3)-B(3,2)*J(2,3))-
(1/p4^2)*(A(4,2)*J(1,4)-B(4,2)*J(2,4));
C(6,3) = (1/p3^2)*(A(3,3)*J(1,3)-B(3,3)*J(2,3))-
(1/p4^2)*(A(4,3)*J(1,4)-B(4,3)*J(2,4));
C(6,4) = (1/p3^2)*(A(3,4)*J(1,3)-B(3,4)*J(2,3))-
(1/p4^2)*(A(4,4)*J(1,4)-B(4,4)*J(2,4));
C(6,5) = (1/p3^2)*(A(3,5)*J(1,3)-B(3,5)*J(2,3))-
(1/p4^2)*(A(4,5)*J(1,4)-B(4,5)*J(2,4));
C(6,6) = (1/p3^2)*(A(3,6)*J(1,3)-B(3,6)*J(2,3))-
(1/p4^2)*(A(4,6)*J(1,4)-B(4,6)*J(2,4));

C(7,1) = ((1/p4^2)*(A(4,1)*J(1,4)-B(4,1)*J(2,4))-(1/pf)*K(1,1));
C(7,2) = ((1/p4^2)*(A(4,2)*J(1,4)-B(4,2)*J(2,4))-(1/pf)*K(1,2));
C(7,3) = ((1/p4^2)*(A(4,3)*J(1,4)-B(4,3)*J(2,4))-(1/pf)*K(1,3));
C(7,4) = ((1/p4^2)*(A(4,4)*J(1,4)-B(4,4)*J(2,4))-(1/pf)*K(1,4));
```

```
C(7,5) = ((1/p4^2)*(A(4,5)*J(1,4)-B(4,5)*J(2,4))-(1/pf)*K(1,5));
C(7,6) = ((1/p4^2)*(A(4,6)*J(1,4)-B(4,6)*J(2,4))-(1/pf)*K(1,6));

C(8,1) = ((1/pf)*K(1,1)-(1/pr)*K(2,1));
C(8,2) = ((1/pf)*K(1,2)-(1/pr)*K(2,2));
C(8,3) = ((1/pf)*K(1,3)-(1/pr)*K(2,3));
C(8,4) = ((1/pf)*K(1,4)-(1/pr)*K(2,4));
C(8,5) = ((1/pf)*K(1,5)-(1/pr)*K(2,5));
C(8,6) = ((1/pf)*K(1,6)-(1/pr)*K(2,6));

C(9,1) = ((1/pr)*K(2,1)-G(3,1));
C(9,2) = ((1/pr)*K(2,2)-G(3,2));
C(9,3) = ((1/pr)*K(2,3)-G(3,3));
C(9,4) = ((1/pr)*K(2,4)-G(3,4));
C(9,5) = ((1/pr)*K(2,5)-G(3,5));
C(9,6) = ((1/pr)*K(2,6)-G(3,6));


%% S simplification terms for dSdq and generalzied force matrices

zikq1_1 = zeros(1,4);
zikq1_1(1) = (-sin(q(5))*cos(q(6))*csc(q(5)+q(6))-Yo(1)/L);
zikq1_1(2) = (-sin(q(5))*cos(q(6))*csc(q(5)+q(6))-Yo(2)/L);
zikq1_1(3) = (-sin(q(5))*cos(q(6))*csc(q(5)+q(6))-Yo(3)/L);
zikq1_1(4) = (-sin(q(5))*cos(q(6))*csc(q(5)+q(6))-Yo(4)/L);

zikq1_2 = zeros(1,4);
zikq1_2(1) = (1-sin(q(5))*cos(q(6))*csc(q(5)+q(6))-(1/2)-Xo(1)/L);
zikq1_2(2) = (1-sin(q(5))*cos(q(6))*csc(q(5)+q(6))-(1/2)-Xo(2)/L);
zikq1_2(3) = (1-sin(q(5))*cos(q(6))*csc(q(5)+q(6))-(1/2)-Xo(3)/L);
zikq1_2(4) = (1-sin(q(5))*cos(q(6))*csc(q(5)+q(6))-(1/2)-Xo(4)/L);

zikq2_1 = zeros(1,4);
zikq2_1(1) = (1+cos(q(5))*cos(q(6))*csc(q(5)+q(6))-(1/2)-Xo(1)/L);
zikq2_1(2) = (1+cos(q(5))*cos(q(6))*csc(q(5)+q(6))-(1/2)-Xo(2)/L);
zikq2_1(3) = (1+cos(q(5))*cos(q(6))*csc(q(5)+q(6))-(1/2)-Xo(3)/L);
zikq2_1(4) = (1+cos(q(5))*cos(q(6))*csc(q(5)+q(6))-(1/2)-Xo(4)/L);

zikq2_2 = zeros(1,4);
zikq2_2(1) = (-cos(q(5))*cos(q(6))*csc(q(5)+q(6))+Yo(1)/L);
zikq2_2(2) = (-cos(q(5))*cos(q(6))*csc(q(5)+q(6))+Yo(2)/L);
zikq2_2(3) = (-cos(q(5))*cos(q(6))*csc(q(5)+q(6))+Yo(3)/L);
zikq2_2(4) = (-cos(q(5))*cos(q(6))*csc(q(5)+q(6))+Yo(4)/L);

zikq3_1 = zeros(1,4);
zikq3_1(1) = (sin(q(5))*cos(q(6))*csc(q(5)+q(6))+Yo(1)/L);
zikq3_1(2) = (sin(q(5))*cos(q(6))*csc(q(5)+q(6))+Yo(2)/L);
zikq3_1(3) = (sin(q(5))*cos(q(6))*csc(q(5)+q(6))+Yo(3)/L);
zikq3_1(4) = (sin(q(5))*cos(q(6))*csc(q(5)+q(6))+Yo(4)/L);

zikq3_2 = zeros(1,4);
zikq3_2(1) = (sin(q(5))*cos(q(6))*csc(q(5)+q(6))-(1/2)+Xo(1)/L);
```

```
zikq3_2(2) = (sin(q(5))*cos(q(6))*csc(q(5)+q(6))-(1/2)+Xo(2)/L);
zikq3_2(3) = (sin(q(5))*cos(q(6))*csc(q(5)+q(6))-(1/2)+Xo(3)/L);
zikq3_2(4) = (sin(q(5))*cos(q(6))*csc(q(5)+q(6))-(1/2)+Xo(4)/L);

zikq4_1 = zeros(1,4);
zikq4_1(1) = (-cos(q(5))*cos(q(6))*csc(q(5)+q(6))-(1/2)+Xo(1)/L);
zikq4_1(2) = (-cos(q(5))*cos(q(6))*csc(q(5)+q(6))-(1/2)+Xo(2)/L);
zikq4_1(3) = (-cos(q(5))*cos(q(6))*csc(q(5)+q(6))-(1/2)+Xo(3)/L);
zikq4_1(4) = (-cos(q(5))*cos(q(6))*csc(q(5)+q(6))-(1/2)+Xo(4)/L);

zikq4_2 = zeros(1,4);
zikq4_2(1) = (cos(q(5))*cos(q(6))*csc(q(5)+q(6))-Yo(1)/L);
zikq4_2(2) = (cos(q(5))*cos(q(6))*csc(q(5)+q(6))-Yo(2)/L);
zikq4_2(3) = (cos(q(5))*cos(q(6))*csc(q(5)+q(6))-Yo(3)/L);
zikq4_2(4) = (cos(q(5))*cos(q(6))*csc(q(5)+q(6))-Yo(4)/L);

zikq5_1 = -((q(1)-q(3))*cos(q(5))+(q(2)-
q(4))*sin(q(5)))*cos(q(6))*csc(q(5)+q(6));
zikq5_2 = ((q(1)-q(3))*sin(q(5))-(q(2)-
q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6));
zikq5_3 = -((q(1)-q(3))*cos(q(5))-(q(2)-
q(4))*sin(q(5)))*cos(q(5))*csc(q(5)+q(6));
zikq5_4 = ((q(1)-q(3))*sin(q(5))+(q(2)-
q(4))*cos(q(5)))*cos(q(6))*csc(q(5)+q(6))*cot(q(5)+q(6));

zikq6_1 = ((q(1)-q(3))*sin(q(5))-(q(2)-
q(4))*cos(q(5)))*(sin(q(6))*csc(q(5)+q(6))+cos(q(6))*csc(q(5)+q(6))*co
t(q(5)+q(6)));
zikq6_2 = ((q(1)-q(3))*sin(q(5))+(q(2)-
q(4))*cos(q(5)))*(sin(q(6))*csc(q(5)+q(6))+cos(q(6))*csc(q(5)+q(6))*co
t(q(5)+q(6)));

%% Generalized Force Matrix
Ti = zeros(4,1);     %Force from Drive Motors
To = zeros(4,1);     %Torque from Steering motors
g = .1; %Tractive Coefficient
Q = zeros(6,4);

for i = 1:4
    Q(1,i) = Ti(i)*((.5+(1-g))*(Xo(i)/L)+((q(2)-q(4))/L^3)*g*((q(2)-
q(4))*Xo(i)+(q(1)-q(3))*Yo(i)))*(-J(2,i)/PR(i))...
           +Ti(i)*((1-g)*(Yo(i)/L)-((q(2)-q(4))/L^3)*g*((q(1)-
q(3))*Xo(i)-(q(2)-q(4))*Yo(i)))*(J(1,i)/PR(i))...
           +(To(i)/PR(i)^2)*(J(1,i)*(-
sin(q(5))*cos(q(6))*csc(q(5)+q(6))-Yo(i)/L)-J(2,i)*(1-
sin(q(5))*cos(q(6))*csc(q(5)+q(6))-.5-Xo(i)/L));

    Q(2,i) = Ti(i)*((-(1-g))*(Yo(i)/L)+((q(1)-q(3))/L^3)*g*((q(2)-
q(4))*Xo(i)+(q(1)-q(3))*Yo(i)))*(-J(2,i)/PR(i))...
           +Ti(i)*((.5+(1-g))*(Yo(i)/L)+((q(1)-q(3))/L^3)*g*((q(1)-
q(3))*Xo(i)-(q(2)-q(4))*Yo(i)))*(J(1,i)/PR(i))...
```

```
+(To(i)/PR(i)^2)*(J(1,i)*(1+cos(q(5))*cos(q(6))*csc(q(5)+q(6))-.5-
Xo(i)/L)-J(2,i)*(-cos(q(5))*cos(q(6))*csc(q(5)+q(6))+Yo(i)/L));

    Q(3,i) = Ti(i)*((.5-(1-g))*(Xo(i)/L)-((q(2)-q(4))/L^3)*g*((q(2)-
q(4))*Xo(i)+(q(1)-q(3))*Yo(i)))*(-J(2,i)/PR(i))...
              +Ti(i)*(((1-g))*(Yo(i)/L)-((q(2)-q(4))/L^3)*g*((q(1)-
q(3))*Xo(i)-(q(2)-q(4))*Yo(i)))*(J(1,i)/PR(i))...

+(To(i)/PR(i)^2)*(J(1,i)*(sin(q(5))*cos(q(6))*csc(q(5)+q(6))+Yo(i)/L)-
J(2,i)*(sin(q(5))*cos(q(6))*csc(q(5)+q(6))-.5+Xo(i)/L));

    Q(4,i) = Ti(i)*(((1-g))*(Yo(i)/L)+((q(1)-q(3))/L^3)*g*((q(2)-
q(4))*Xo(i)+(q(1)-q(3))*Yo(i)))*(-J(2,i)/PR(i))...
              +Ti(i)*((.5-(1-g))*(Xo(i)/L)-((q(1)-q(3))/L^3)*g*((q(1)-
q(3))*Xo(i)-(q(2)-q(4))*Yo(i)))*(J(1,i)/PR(i))...
              +(To(i)/PR(i)^2)*(J(1,i)*(-
cos(q(5))*cos(q(6))*csc(q(5)+q(6))-.5+Xo(i)/L)-
J(2,i)*(cos(q(5))*cos(q(6))*csc(q(5)+q(6))-Yo(i)/L));

    Q(5,i) = (To(i)/PR(i)^2)*((J(1,i)*(zikq5_1+zikq5_2))-
J(2,i)*(zikq5_3+zikq5_4));
    Q(6,i) = (To(i)/PR(i)^2)*(J(1,i)*(zikq6_1)-J(2,i)*zikq6_2);
end


%% RO derivatives for q5 q6 from PR1 PR2 PR3 PR4

sq1 =
sqrt(W^2*sin(q(5)+q(6))^2/(4*L^2*cos(q(6))^2)+W*sin(q(5)+q(6))*cos(q(5
))/(L*cos(q(6)))+1);
sq2 = sqrt(W^2*sin(q(5)+q(6))^2/(4*L^2*cos(q(6))^2)-
W*sin(q(5)+q(6))*cos(q(5))/(L*cos(q(6)))+1);
sq3 = sqrt(W^2*sin(q(5)+q(6))^2/(4*L^2*cos(q(5))^2)-
W*sin(q(5)+q(6))*cos(q(6))/(L*cos(q(5)))+1);
sq4 =
sqrt(W^2*sin(q(5)+q(6))^2/(4*L^2*cos(q(5))^2)+W*sin(q(5)+q(6))*cos(q(6
))/(L*cos(q(5)))+1);

%derivative of ro wrt q5
PRd = zeros(4,2);
PRd(1,1) = L*cos(q(6))*((W*cos(q(5)+q(6))*cos(q(5))/(L*cos(q(6))))-
(W*sin(q(5)+q(6))*sin(q(5))/(L*cos(q(6))))+(W^2*cos(q(5)+q(6))*sin(q(5
)+q(6))/(2*L^2*cos(q(6))^2)))/(2*sin(q(5)+q(6))*sq1)-
(L*cos(q(5)+q(6))*cos(q(6))*sq1)/sin(q(5)+q(6))^2; %
PRd(2,1)=  L*cos(q(6))*((W*sin(q(5)+q(6))*sin(q(5))/(L*cos(q(6))))-
(W*cos(q(5)+q(6))*cos(q(5))/(L*cos(q(6))))+(W^2*cos(q(5)+q(6))*sin(q(5
)+q(6))/(2*L^2*cos(q(6))^2)))/(2*sin(q(5)+q(6))*sq2)-
(L*cos(q(5)+q(6))*cos(q(6))*sq2)/(sin(q(5)+q(6)))^2; %
PRd(3,1)= -L*sin(q(5))*sq3/sin(q(5)+q(6))-
(L*cos(q(5)+q(6))*cos(q(5))*sq3/sin(q(5)+q(6))^2)-
L*cos(q(5))*((W*cos(q(5)+q(6))*cos(q(6))/(L*cos(q(5))))-
```

```
(W^2*cos(q(5)+q(6))*sin(q(5)+q(6))/(2*L^2*cos(q(5))^2))-
(W^2*sin(q(5)+q(6))^2*sin(q(5))/(2*L^2*cos(q(5))^3))+W*sin(q(5)+q(6))*
cos(q(6))*sin(q(5))/(L*cos(q(5))^2))/(2*sin(q(5)+q(6))*sq3);%
PRd(4,1)= -(L*cos(q(5)+q(6))*cos(q(5))*sq4/sin(q(5)+q(6))^2)-
L*sin(q(5))*sq4/sin(q(5)+q(6))+L*cos(q(5))*((W*cos(q(5)+q(6))*cos(q(6)
)/(L*cos(q(5))))+(W^2*cos(q(5)+q(6))*sin(q(5)+q(6))/(2*L^2*cos(q(5))^2
))+(W^2*sin(q(5)+q(6))^2*sin(q(5))/(2*L^2*cos(q(5))^3))+W*sin(q(5)+q(6
))*cos(q(6))*sin(q(5))/(L*cos(q(5))^2))/(2*sin(q(5)+q(6))*sq4);%

%derivative of ro wrt q6
PRd(1,2) =
L*cos(q(6))*((W*cos(q(5)+q(6))*cos(q(5))/(L*cos(q(6))))+(W^2*cos(q(5)+
q(6))*sin(q(5)+q(6))/(2*L^2*cos(q(6))^2))+(W^2*sin(q(5)+q(6))^2*sin(q(
6))/(2*L^2*cos(q(6))^3))+(W*sin(q(5)+q(6))*cos(q(5))*sin(q(6))/(L*cos(
q(6))^2)))/(2*sin(q(5)+q(6))*sq1)-
(L*cos(q(5)+q(6))*cos(q(6))*sq1)/sin(q(5)+q(6))^2-
L*sin(q(6))*sq1/sin(q(5)+q(6));%
PRd(2,2) = -L*cos(q(6))*((W*cos(q(5)+q(6))*cos(q(5))/(L*cos(q(6))))-
(W^2*cos(q(5)+q(6))*sin(q(5)+q(6))/(2*L^2*cos(q(6))^2))-
(W^2*sin(q(5)+q(6))^2*sin(q(6))/(2*L^2*cos(q(6))^3))+(W*sin(q(5)+q(6))
*cos(q(5))*sin(q(6))/(L*cos(q(6))^2)))/(2*sin(q(5)+q(6))*sq2)-
(L*cos(q(5)+q(6))*cos(q(6))*sq2)/sin(q(5)+q(6))^2-
L*sin(q(6))*sq2/sin(q(5)+q(6));%
PRd(3,2) =  L*cos(q(5))*((W*sin(q(5)+q(6))*sin(q(6))/(L*cos(q(5))))-
(W*cos(q(5)+q(6))*cos(q(6))/(L*cos(q(5))))+(W^2*cos(q(5)+q(6))*sin(q(5
)+q(6))/(2*L^2*cos(q(5))^2)))/(2*sin(q(5)+q(6))*sq3)-
(L*cos(q(5)+q(6))*cos(q(5))*sq3)/sin(q(5)+q(6))^2; %
PRd(4,2) =  L*cos(q(5))*((W*cos(q(5)+q(6))*cos(q(6))/(L*cos(q(5))))-
(W*sin(q(5)+q(6))*sin(q(6))/(L*cos(q(5))))+(W^2*cos(q(5)+q(6))*sin(q(5
)+q(6))/(2*L^2*cos(q(5))^2)))/(2*sin(q(5)+q(6))*sq4)-
(L*cos(q(5)+q(6))*cos(q(5))*sq4)/sin(q(5)+q(6))^2; %



%% dSdq Matrix, S derivative

dSdq = zeros(4,6,6); %i is the 4, k is the column, j is the depth


for i = 1:4
    for k = 1:6
        dSdq(i,k,1) = (zikq1_1(i)*(A(i,k)-H(1,k))-zikq1_2(i)*(B(i,k)-
H(2,k)))/PR(i)^2;
        dSdq(i,k,2) = (zikq2_1(i)*(A(i,k)-H(1,k))-zikq2_2(i)*(B(i,k)-
H(2,k)))/PR(i)^2;
        dSdq(i,k,3) = (zikq3_1(i)*(A(i,k)-H(1,k))-zikq3_2(i)*(B(i,k)-
H(2,k)))/PR(i)^2;
        dSdq(i,k,4) = (zikq4_1(i)*(A(i,k)-H(1,k))-zikq4_2(i)*(B(i,k)-
H(2,k)))/PR(i)^2;
        dSdq(i,k,5) = (PR(i)^2*((zikq5_1+zikq5_2)*(A(i,k)-
H(1,k))+J(2,i)*Hx(k,5)-J(1,i)*Hy(k,5)-(zikq5_3+zikq5_4)*(B(i,k)-
```

```
H(2,k)))-(J(2,i)*(A(i,k)-H(1,k))-J(1,i)*(B(i,k)-
H(2,k)))*2*PR(i)*PRd(i,1))/PR(i)^4;
       dSdq(i,k,6) = (PR(i)^2*((zikq6_1)*(A(i,k)-
H(1,k))+J(2,i)*Hx(k,6)-J(1,i)*Hy(k,6)-(zikq6_2)*(B(i,k)-H(2,k)))-
(J(2,i)*(A(i,k)-H(1,k))-J(1,i)*(B(i,k)-
H(2,k)))*2*PR(i)*PRd(i,2))/PR(i)^4;
    end
end
%% Term Colation


Tp1_1equation = zeros(6,6); % THis sum is for the mB portion of the KE
in Equation (2.158)
Tp1_2equation = zeros(6,6); % THis sum is for the IB portion of the KE
in Equation (2.158)
Tp1_3equation = zeros(6,6,4); % THis sum is for the (mw+Iw/rw^2)
portion of the KE in Equation (2.158)
Tp1_4equation = zeros(6,6,4); % THis sum is for the Is portion of the
KE in Equation (2.158)

% This summation is for the first two lines of Equation 2.158
for j = 1:6
    for k=1:6
         Tp1_1equation(j,k) = G(1,j)*G(1,k)+G(2,j)*G(2,k);
         Tp1_2equation(j,k) = G(3,j)*G(3,k);

         for i=1:4
            Tp1_3equation(j,k,i) = A(i,j)*A(i,k)+B(i,j)*B(i,k);
            Tp1_4equation(j,k,i) = S(i,j)*S(i,k);
         end
    end
end

EquationPart1_1 =
Mb*sum(Tp1_1equation(1,:))*qdd(1)+Ib*sum(Tp1_2equation(1,:))*qdd(1)+(M
i+Ii/rw^2)*sum(sum(Tp1_3equation(1,:,:)))*qdd(1)+(Is)*sum(sum(Tp1_4equ
ation(1,:,:)))*qdd(1);
EquationPart1_2 =
Mb*sum(Tp1_1equation(2,:))*qdd(2)+Ib*sum(Tp1_2equation(2,:))*qdd(1)+(M
i+Ii/rw^2)*sum(sum(Tp1_3equation(2,:,:)))*qdd(2)+(Is)*sum(sum(Tp1_4equ
ation(1,:,:)))*qdd(2);
EquationPart1_3 =
Mb*sum(Tp1_1equation(3,:))*qdd(3)+Ib*sum(Tp1_2equation(3,:))*qdd(1)+(M
i+Ii/rw^2)*sum(sum(Tp1_3equation(3,:,:)))*qdd(3)+(Is)*sum(sum(Tp1_4equ
ation(1,:,:)))*qdd(3);
EquationPart1_4 =
Mb*sum(Tp1_1equation(4,:))*qdd(4)+Ib*sum(Tp1_2equation(4,:))*qdd(1)+(M
i+Ii/rw^2)*sum(sum(Tp1_3equation(4,:,:)))*qdd(4)+(Is)*sum(sum(Tp1_4equ
ation(1,:,:)))*qdd(4);
EquationPart1_5 =
Mb*sum(Tp1_1equation(5,:))*qdd(5)+Ib*sum(Tp1_2equation(5,:))*qdd(1)+(M
```

```
i+Ii/rw^2)*sum(sum(Tp1_3equation(5,:,:)))*qdd(5)+(Is)*sum(sum(Tp1_4equation(1,:,:)))*qdd(5);
EquationPart1_6 =
Mb*sum(Tp1_1equation(6,:))*qdd(6)+Ib*sum(Tp1_2equation(6,:))*qdd(1)+(Mi+Ii/rw^2)*sum(sum(Tp1_3equation(6,:,:)))*qdd(6)+(Is)*sum(sum(Tp1_4equation(1,:,:)))*qdd(6);

SUMS_KNI1 = zeros(6,6,4);   %This matrix is for the first triple summation in Equation 2.158
SUMS_KNI2 = zeros(6,6,4);   %This matrix is for the second triple summation in Equation 2.158

%here I am building the terms through the summation shown in lines 3 and 4
%of Equation 2.158
for k=1:6
    for n=1:6
        for i=1:4
            SUMS_KNI1(k,n,i) = S(i,1)*sum(dSdq(i,k,n)) + S(i,k)'*sum(dSdq(i,1,n))  ;%EQ1
            SUMS_KNI1(k,n,i) = S(i,2)*sum(dSdq(i,k,n)) + S(i,k)'*sum(dSdq(i,2,n))  ;%EQ2
            SUMS_KNI1(k,n,i) = S(i,3)*sum(dSdq(i,k,n)) + S(i,k)'*sum(dSdq(i,3,n))  ;%EQ3
            SUMS_KNI1(k,n,i) = S(i,4)*sum(dSdq(i,k,n)) + S(i,k)'*sum(dSdq(i,4,n))  ;%EQ4
            SUMS_KNI1(k,n,i) = S(i,5)*sum(dSdq(i,k,n)) + S(i,k)'*sum(dSdq(i,5,n))  ;%EQ5
            SUMS_KNI1(k,n,i) = S(i,6)*sum(dSdq(i,k,n)) + S(i,k)'*sum(dSdq(i,6,n))  ;%EQ6

            SUMS_KNI2(k,n,i) = S(i,k)*sum(dSdq(i,k,1)) + S(i,k)'*sum(dSdq(i,n,1))  ;%EQ1
            SUMS_KNI2(k,n,i) = S(i,k)*sum(dSdq(i,k,2)) + S(i,k)'*sum(dSdq(i,n,2))  ;%EQ2
            SUMS_KNI2(k,n,i) = S(i,k)*sum(dSdq(i,k,3)) + S(i,k)'*sum(dSdq(i,n,3))  ;%EQ3
            SUMS_KNI2(k,n,i) = S(i,k)*sum(dSdq(i,k,4)) + S(i,k)'*sum(dSdq(i,n,4))  ;%EQ4
            SUMS_KNI2(k,n,i) = S(i,k)*sum(dSdq(i,k,5)) + S(i,k)'*sum(dSdq(i,n,5))  ;%EQ5
            SUMS_KNI2(k,n,i) = S(i,k)*sum(dSdq(i,k,6)) + S(i,k)'*sum(dSdq(i,n,6))  ;%EQ6
        end
    end
end

SUMS_kni1_Total = zeros(6,6);
SUMS_kni2_Total = zeros(6,6);
for i = 6
    for j = 1:6
        SUMS_kni1_Total(i,j) = sum(SUMS_KNI1(i,j,:))*qd(i)*qd(j);
```

```
        SUMS_kni2_Total(i,j) = sum(SUMS_KNI2(i,j,:))*qd(i)*qd(j);
    end
end

%here I am summing the terms of the summation shown in lines 3 and 4
%of Equation 2.158
EquationPart2_1 = Is*SUMS_kni1_Total(1,:)-
(1/2)*Is*SUMS_kni2_Total(1,:);
EquationPart2_2 = Is*SUMS_kni1_Total(2,:)-
(1/2)*Is*SUMS_kni2_Total(2,:);
EquationPart2_3 = Is*SUMS_kni1_Total(3,:)-
(1/2)*Is*SUMS_kni2_Total(3,:);
EquationPart2_4 = Is*SUMS_kni1_Total(4,:)-
(1/2)*Is*SUMS_kni2_Total(4,:);
EquationPart2_5 = Is*SUMS_kni1_Total(5,:)-
(1/2)*Is*SUMS_kni2_Total(5,:);
EquationPart2_6 = Is*SUMS_kni1_Total(6,:)-
(1/2)*Is*SUMS_kni2_Total(6,:);

%This is the left hand side of the EOM. (d/dt)*(dL/dqd)-dL/dq

Lagrangian_EOM_Left_Side_1 = EquationPart1_1 + EquationPart2_1;
Lagrangian_EOM_Left_Side_2 = EquationPart1_2 + EquationPart2_2;
Lagrangian_EOM_Left_Side_3 = EquationPart1_3 + EquationPart2_3;
Lagrangian_EOM_Left_Side_4 = EquationPart1_4 + EquationPart2_4;
Lagrangian_EOM_Left_Side_5 = EquationPart1_5 + EquationPart2_5;
Lagrangian_EOM_Left_Side_6 = EquationPart1_6 + EquationPart2_6;


Q1 = sum(Q(1,:));   %Generalized Force 1
Q2 = sum(Q(2,:));   %Generalized Force 2
Q3 = sum(Q(3,:));   %Generalized Force 3
Q4 = sum(Q(4,:));   %Generalized Force 4
Q5 = sum(Q(5,:));   %Generalized Force 5
Q6 = sum(Q(6,:));   %Generalized Force 6

% The constraints are above, but will be summd here

C1 = sum(C(1,:));
C2 = sum(C(2,:));
C3 = sum(C(3,:));
C4 = sum(C(4,:));
C5 = sum(C(5,:));
C6 = sum(C(6,:));
C7 = sum(C(7,:));
C8 = sum(C(8,:));
C9 = sum(C(9,:));
```

A.1.2. Simulation.m

```
MyConstants
    y=[0;0;0;0;0;0;0;0;0;0;0;0];
    dt=0.005
    Px=0:1:1000;
    Py=3*sin(2*pi*Px/400);
    for s=1:1:1000
        Ti = 5e-2*[1;1;1;1];     %Force from Drive Motors
        To = 0.2*sin(2*pi*s/1000)*[-1,1,-1,1];
       % To = 0.3*sin(2*pi*s/1000)*[-1,-1,1,1];
        %To = 0.5*sin(2*pi*s/1000)*[1,1,-1,-1];
        Equations_Non_Symbolic;
        Mm1=Mb*(G(1,:)'*G(1,:)+G(2,:)'*G(2,:))+Ib*G(3,:)'*G(3,:);
        Mm2=(Mi+Ii/rw^2)*(A'*A+B'*B)+Is*S'*S;
        Mm=Mm1+Mm2;
        Sdsdq1=2*S'*theDsDq;
        X=[q;qd];

F=[X(7);X(8);X(9);X(10);X(11);X(12);pinv(Mm)*(Sdsdq1)*[X(1);X(2);X(3);
X(4);X(5);X(6)]+[Q1;Q2;Q3;Q4;Q5;Q6]+C'*Lamb'*1e-10];
        if s>2
            X=y(:,s)+dt*(y(:,s-1)+F)/2;
        else
            X=y(:,s)+dt*F;
        end
        for i=1:6
            q(i)=X(i);
        end
        for i=1:6
            qd(i)=X(i+6);
        end
        y=[y,X];
    end
    y=y*10/dt;
    figure(1)
%    plot(y(1,:),y(7,:),'r-.',y(2,:),y(8,:),'k-
',y(3,:),y(9,:),'b:',y(4,:),y(10,:),'g--
')%,y(5,:),y(11,:),'m+',y(6,:),y(12,:),'c+')
    plot(s,y(7,:),'r-.',s,y(8,:),'k-',s,y(9,:),'b:',s,y(10,:),'g--
')%,y(5,:),y(11,:),'m+',y(6,:),y(12,:),'c+')
    grid
    xlabel('positions')
    ylabel('Velocities')
    legend('Wheel 1', 'Wheel 2','Wheel 3','Wheel 4')
```

A.2. H Files for Experimental Prototype

A.2.1. RTC_Initialization.h

```
#ifndef KEK__RTC__Initializations
#define KEK__RTC__Initializations

#include "tm_stm32f4_rtc.h"




#endif
```

A.2.2. Pdm_filter.h

```
/* Define to prevent recursive inclusion ------------------------
-------------*/
#ifndef __PDM_FILTER_H
#define __PDM_FILTER_H

#ifdef __cplusplus
 extern "C" {
#endif

/* Includes -----------------------------------------------------
-------------*/
#include <stdint.h>

/* Exported types -----------------------------------------------
-------------*/
typedef struct {
    uint16_t Fs;
    float LP_HZ;
    float HP_HZ;
    uint16_t In_MicChannels;
    uint16_t Out_MicChannels;
    char InternalFilter[34];
} PDMFilter_InitStruct;

/* Exported constants -------------------------------------------
-------------*/
/* Exported macros ----------------------------------------------
-------------*/
#define HTONS(A)  ((((u16)(A) & 0xff00) >> 8) | \
                   (((u16)(A) & 0x00ff) << 8))

/* Exported functions -------------------------------------------
------------ */
void PDM_Filter_Init(PDMFilter_InitStruct * Filter);
```

```
	int32_t PDM_Filter_64_MSB(uint8_t* data, uint16_t* dataOut, 
uint16_t MicGain,  PDMFilter_InitStruct * Filter);
	int32_t PDM_Filter_80_MSB(uint8_t* data, uint16_t* dataOut, 
uint16_t MicGain,  PDMFilter_InitStruct * Filter);
	int32_t PDM_Filter_64_LSB(uint8_t* data, uint16_t* dataOut, 
uint16_t MicGain,  PDMFilter_InitStruct * Filter);
	int32_t PDM_Filter_80_LSB(uint8_t* data, uint16_t* dataOut, 
uint16_t MicGain,  PDMFilter_InitStruct * Filter);

	#ifdef __cplusplus
	}
	#endif

	#endif /* __PDM_FILTER_H */
```

A.2.3. DC_Motor_Initializations.h

```
	#ifndef _KARL_DC_Motor_Initializations_H_
	#define _KARL_DC_Motor_Initializations_H_

	#include <stm32f4xx_rcc.h>
	#include <stm32f4xx_tim.h>

	#include "tm_stm32f4_gpio.h"    // General Purpose Input/Output
	#include "tm_stm32f4_spi.h"     // Serial Peripheral Interface


	void DC_GPIO_Initializations(void) {
	GPIO_InitTypeDef GPIO_InitStructure;

		/*________________INITIALIZE PERIPHERAL 
CLOCK______________*/

		RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

		/*Timer Initializations*/

		/*DC 1 | DC 2* | DC 3 | DC4/

		/* GPIOE Configuration: TIM4 CH1 (PD12), TIM4 CH2 (PD13) 
TIM4 CH3 (PD14), TIM4 CH4 (PD15) */

		GPIO_InitStructure.GPIO_Pin=GPIO_Pin_12 | GPIO_Pin_13 | 
GPIO_Pin_14 | GPIO_Pin_15 ;
		GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AF;
		GPIO_InitStructure.GPIO_OType=GPIO_OType_PP;
		GPIO_InitStructure.GPIO_Speed=GPIO_Speed_100MHz;
		GPIO_InitStructure.GPIO_PuPd=GPIO_PuPd_UP;
		GPIO_Init(GPIOD,&GPIO_InitStructure);
```

```
        GPIO_PinAFConfig(GPIOD,GPIO_PinSource12,GPIO_AF_TIM4);
        GPIO_PinAFConfig(GPIOD,GPIO_PinSource13,GPIO_AF_TIM4);
        GPIO_PinAFConfig(GPIOD,GPIO_PinSource14,GPIO_AF_TIM4);
        GPIO_PinAFConfig(GPIOD,GPIO_PinSource15,GPIO_AF_TIM4);
    }


    void DC_TIM_Init(void) {


        uint32_t CCR1_Val = 0;
        uint32_t CCR2_Val = 0;
        uint32_t CCR3_Val = 0;
        uint32_t CCR4_Val = 0;

        GPIO_PinAFConfig(GPIOD,GPIO_PinSource12,GPIO_AF_TIM4);
        GPIO_PinAFConfig(GPIOD,GPIO_PinSource13,GPIO_AF_TIM4);
        GPIO_PinAFConfig(GPIOD,GPIO_PinSource14,GPIO_AF_TIM4);
        GPIO_PinAFConfig(GPIOD,GPIO_PinSource15,GPIO_AF_TIM4);
        TIM_TimeBaseInitTypeDef TIM_BaseStruct;


            /*Clock for TIM4 */
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);

        TIM_BaseStruct.TIM_Prescaler = 6;
        TIM_BaseStruct.TIM_CounterMode = TIM_CounterMode_Up; /*
Count up */
        TIM_BaseStruct.TIM_Period = 60000; /*The ARR value for
200HZ*/
        TIM_BaseStruct.TIM_ClockDivision = TIM_CKD_DIV1;
        TIM_BaseStruct.TIM_RepetitionCounter = 0;
        TIM_TimeBaseInit(TIM4, &TIM_BaseStruct); /* Initialize TIM4
*/
        TIM_Cmd(TIM4, ENABLE); /* Start count on TIM4 */

        TIM_OCInitTypeDef TIM_OCStruct;
            /* OC Settings */
        TIM_OCStruct.TIM_OCMode = TIM_OCMode_PWM1;
        TIM_OCStruct.TIM_OutputState = TIM_OutputState_Enable;
        TIM_OCStruct.TIM_OCPolarity = TIM_OCPolarity_High;

            /*Set toggle period for each channel*/
        TIM_OCStruct.TIM_Pulse = CCR1_Val; /* 25% Toggle */
        TIM_OC1Init(TIM4, &TIM_OCStruct);
        TIM_OC1PreloadConfig(TIM4, TIM_OCPreload_Enable);

        TIM_OCStruct.TIM_Pulse = CCR2_Val; /* 50% Toggle */
        TIM_OC2Init(TIM4, &TIM_OCStruct);
        TIM_OC2PreloadConfig(TIM4, TIM_OCPreload_Enable);

        TIM_OCStruct.TIM_Pulse = CCR3_Val; /* 75% Toggle */
```

```
        TIM_OC3Init(TIM4, &TIM_OCStruct);
        TIM_OC3PreloadConfig(TIM4, TIM_OCPreload_Enable);

        TIM_OCStruct.TIM_Pulse = CCR4_Val; /* 100% Toggle */
        TIM_OC4Init(TIM4, &TIM_OCStruct);
        TIM_OC4PreloadConfig(TIM4, TIM_OCPreload_Enable);
    }


    /*
     *   TM_GPIO_Init(
                    //DC Motor Pins
                    GPIOD,
                    // GPIO Port D
                    GPIO_Pin_12 | GPIO_Pin_13 | GPIO_Pin_14 |
GPIO_Pin_15,
                    TM_GPIO_Mode_AF,
                // Mode: output
                    TM_GPIO_OType_PP,
                    // Mode: Push/Pull
                    TM_GPIO_PuPd_UP,
                    // No pull up/down resistor
                    TM_GPIO_Speed_Fast
                // ...fast
            );
            */

    #endif // _KARL_DC_Motor_Initializations_H_
```

A.2.4. DC_Motor_Actuation.h

```
    #ifndef _KARL_DC_Motor_Actuation_H_
    #define _KARL_DC_Motor_Actuation_H_

    uint32_t CCR1_Val = 18000;
    uint32_t CCR2_Val = 18000;
    uint32_t CCR3_Val = 18000;
    uint32_t CCR4_Val = 18000;

    void DC_Motor_Actuate(float* Power)
    {
        int i;
        //uint32_t CCR[4] = {21000,12000,24000,12000};
        //*waypoint = 1;

      /*CCR1_Val = 6000/100*Power[0]+18000;
        CCR2_Val = 6000/100*Power[1]+18000;
        CCR3_Val = 6000/100*Power[2]+18000;
        CCR4_Val = 6000/100*Power[3]+18000;
      */
```

```
        CCR2_Val = 6000/100*Power[0]+18000;
        CCR3_Val = 6000/100*Power[2]+18000;
        CCR1_Val = 6000/100*Power[1]+18000;
        CCR4_Val = 6000/100*Power[3]+18000;


        TIM4->CCR1 = CCR4_Val;      //Motor 4
        TIM4->CCR2 = CCR2_Val;      //Motor 3
        TIM4->CCR3 = CCR3_Val;      //Motor 2
        TIM4->CCR4 = CCR1_Val;      //Motor 1
    }

    #endif //_KARL_DC_Motor_Actuation_H_
```

A.2.5. DC_Motor_PID.h

```
    #ifndef    DC_MOTOR_PID_CONTROLLER_KEK
    #define DC_MOTOR_PID_CONTROLLER_KEK

    void DC_MOTOR_PID(float* Power, float* VelocityError, float*
IntegralError){
        float Kp = 0;
        float Ki = .1;
        float PowerOld[4] = {Power[0],Power[1],Power[2],Power[3]};
        float dt =  1;


        IntegralError[0] = IntegralError[0] + VelocityError[0]*dt;
        IntegralError[1] = IntegralError[1] + VelocityError[1]*dt;
        IntegralError[2] = IntegralError[2] + VelocityError[2]*dt;
        IntegralError[3] = IntegralError[3] + VelocityError[3]*dt;

        Power[0] = Ki*IntegralError[0] + Kp*VelocityError[0];
        Power[1] = Ki*IntegralError[1] + Kp*VelocityError[1];
        Power[2] = Ki*IntegralError[2] + Kp*VelocityError[2];
        Power[3] = Ki*IntegralError[3] + Kp*VelocityError[3];


    //    Power[0] = 2*Kp*(VelocityError[0]+PowerOld[0]  +
0*VelocityError[0]);
    //    Power[1] = Kp*VelocityError[1]+PowerOld[1];
    //    Power[2] = Kp*VelocityError[2]+PowerOld[2];
    //    Power[3] = Kp*VelocityError[3]+PowerOld[3];

    }


    #endif     //DC_MOTOR_PID_CONTROLLER_KEK
```

A.2.6. Stepper_Initializations.h

```
#ifndef __KEK__Stepper___INITIALIZATION__H__
#define __KEK__Stepper___INITIALIZATION__H__

void Stepper_GPIO_Initialization(void){

      //Initialize Direction pins for the four motors
      TM_GPIO_Init(
                          GPIOD,
                        // GPIO Port D
                          GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 |
GPIO_Pin_3,         // Pin 0, 1, 2, 3
                          TM_GPIO_Mode_OUT,
                        // Mode: output
                          TM_GPIO_OType_PP,
                        // Mode: Push/Pull
                          TM_GPIO_PuPd_NOPULL,
                        // No pull up/down resistor
                          TM_GPIO_Speed_Fast   // ...fast
                 );

      //Initialize On/Off pins for the four motors
      TM_GPIO_Init(
                          GPIOE,
                        // GPIO Port E
                          GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 |
GPIO_Pin_3,         // Pin 0, 1, 2, 3
                          TM_GPIO_Mode_OUT,
                        // Mode: output
                          TM_GPIO_OType_PP,
                        // Mode: Push/Pull
                          TM_GPIO_PuPd_NOPULL,
                        // No pull up/down resistor
                          TM_GPIO_Speed_Fast   // ...fast
                 );

      GPIO_InitTypeDef GPIO_InitStructure;

      RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOE, ENABLE);

      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;
      GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AF;
      GPIO_InitStructure.GPIO_OType=GPIO_OType_PP;
      GPIO_InitStructure.GPIO_Speed=GPIO_Speed_100MHz;
      GPIO_InitStructure.GPIO_PuPd=GPIO_PuPd_UP;
      GPIO_Init(GPIOE,&GPIO_InitStructure);

      GPIO_PinAFConfig(GPIOE,GPIO_PinSource6,GPIO_AF_TIM9);
```

```
          RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);

          GPIO_InitStructure.GPIO_Pin=GPIO_Pin_8 | GPIO_Pin_9|
GPIO_Pin_14;
          GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AF;
          GPIO_InitStructure.GPIO_OType=GPIO_OType_PP;
          GPIO_InitStructure.GPIO_Speed=GPIO_Speed_100MHz;
          GPIO_InitStructure.GPIO_PuPd=GPIO_PuPd_UP;
          GPIO_Init(GPIOB,&GPIO_InitStructure);

          GPIO_PinAFConfig(GPIOB,GPIO_PinSource8, GPIO_AF_TIM10);
          GPIO_PinAFConfig(GPIOB,GPIO_PinSource9, GPIO_AF_TIM11);
          GPIO_PinAFConfig(GPIOB,GPIO_PinSource14,GPIO_AF_TIM12);
     }

     void Stepper_TIM_9_Init(void) {
          uint32_t CCR2_Per = 0;


          /*Clock for TIM9 */
          RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM9, ENABLE);


          TIM_TimeBaseInitTypeDef TIM_BaseStruct;

          TIM_BaseStruct.TIM_Prescaler = 6;
          TIM_BaseStruct.TIM_CounterMode = TIM_CounterMode_Up; /*
Count up */
          TIM_BaseStruct.TIM_Period = CCR2_Per; /*The ARR value for
200HZ*/
          TIM_BaseStruct.TIM_ClockDivision = TIM_CKD_DIV1;
          TIM_BaseStruct.TIM_RepetitionCounter = 0;
          TIM_TimeBaseInit(TIM9, &TIM_BaseStruct); /* Initialize TIM4
*/
          TIM_Cmd(TIM9, ENABLE); /* Start count on TIM4 */

          TIM_OCInitTypeDef TIM_OCStruct;
               /* OC Settings */
          TIM_OCStruct.TIM_OCMode = TIM_OCMode_PWM1;
          TIM_OCStruct.TIM_OutputState = TIM_OutputState_Enable;
          TIM_OCStruct.TIM_OCPolarity = TIM_OCPolarity_High;

          TIM_OCStruct.TIM_Pulse = CCR2_Per/2; /* 50% Toggle */
          TIM_OC2Init(TIM9, &TIM_OCStruct);
          TIM_OC2PreloadConfig(TIM9, TIM_OCPreload_Enable);

     }

     void Stepper_TIM_10_Init(void) {
          uint32_t CCR1_Per = 0;
```

```
        /*Clock for TIM10 */
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM10, ENABLE);

        TIM_TimeBaseInitTypeDef TIM_BaseStruct;

        TIM_BaseStruct.TIM_Prescaler = 6;
        TIM_BaseStruct.TIM_CounterMode = TIM_CounterMode_Up; /*
Count up */
        TIM_BaseStruct.TIM_Period = CCR1_Per/2; /*The ARR value for
200HZ*/
        TIM_BaseStruct.TIM_ClockDivision = TIM_CKD_DIV1;
        TIM_BaseStruct.TIM_RepetitionCounter = 0;
        TIM_TimeBaseInit(TIM10, &TIM_BaseStruct); /* Initialize
TIM4 */
        TIM_Cmd(TIM10, ENABLE); /* Start count on TIM4 */

        TIM_OCInitTypeDef TIM_OCStruct;
            /* OC Settings */
        TIM_OCStruct.TIM_OCMode = TIM_OCMode_PWM1;
        TIM_OCStruct.TIM_OutputState = TIM_OutputState_Enable;
        TIM_OCStruct.TIM_OCPolarity = TIM_OCPolarity_High;

            /*Set toggle period for each channel*/
        TIM_OCStruct.TIM_Pulse = CCR1_Per/4; /* 25% Toggle */
        TIM_OC1Init(TIM10, &TIM_OCStruct);
        TIM_OC1PreloadConfig(TIM10, TIM_OCPreload_Enable);
    }

    void Stepper_TIM_11_Init(void) {
        uint32_t CCR1_Per = 0;

        /*Clock for TIM11 */
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM11, ENABLE);

        TIM_TimeBaseInitTypeDef TIM_BaseStruct;

        TIM_BaseStruct.TIM_Prescaler = 6;
        TIM_BaseStruct.TIM_CounterMode = TIM_CounterMode_Up; /*
Count up */
        TIM_BaseStruct.TIM_Period = CCR1_Per/2; /*The ARR value for
200HZ*/
        TIM_BaseStruct.TIM_ClockDivision = TIM_CKD_DIV1;
        TIM_BaseStruct.TIM_RepetitionCounter = 0;
        TIM_TimeBaseInit(TIM11, &TIM_BaseStruct); /* Initialize
TIM4 */
        TIM_Cmd(TIM11, ENABLE); /* Start count on TIM4 */

        TIM_OCInitTypeDef TIM_OCStruct;
            /* OC Settings */
        TIM_OCStruct.TIM_OCMode = TIM_OCMode_PWM1;
        TIM_OCStruct.TIM_OutputState = TIM_OutputState_Enable;
        TIM_OCStruct.TIM_OCPolarity = TIM_OCPolarity_High;
```

```
            /*Set toggle period for each channel*/
        TIM_OCStruct.TIM_Pulse = CCR1_Per/4; /* 25% Toggle */
        TIM_OC1Init(TIM11, &TIM_OCStruct);
        TIM_OC1PreloadConfig(TIM11, TIM_OCPreload_Enable);
    }

    void Stepper_TIM_12_Init(void) {
        uint32_t CCR1_Per = 0;

        /*Clock for TIM12 */
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM12, ENABLE);

        TIM_TimeBaseInitTypeDef TIM_BaseStruct;

        TIM_BaseStruct.TIM_Prescaler = 6;
        TIM_BaseStruct.TIM_CounterMode = TIM_CounterMode_Up; /*
Count up */
        TIM_BaseStruct.TIM_Period = CCR1_Per; /*The ARR value for
200HZ*/
        TIM_BaseStruct.TIM_ClockDivision = TIM_CKD_DIV1;
        TIM_BaseStruct.TIM_RepetitionCounter = 0;
        TIM_TimeBaseInit(TIM12, &TIM_BaseStruct); /* Initialize
TIM4 */
        TIM_Cmd(TIM12, ENABLE); /* Start count on TIM4 */

        TIM_OCInitTypeDef TIM_OCStruct;
            /* OC Settings */
        TIM_OCStruct.TIM_OCMode = TIM_OCMode_PWM1;
        TIM_OCStruct.TIM_OutputState = TIM_OutputState_Enable;
        TIM_OCStruct.TIM_OCPolarity = TIM_OCPolarity_High;

            /*Set toggle period for each channel*/
        TIM_OCStruct.TIM_Pulse = CCR1_Per/2; /* 25% Toggle */
        TIM_OC1Init(TIM12, &TIM_OCStruct);
        TIM_OC1PreloadConfig(TIM12, TIM_OCPreload_Enable);
    }
    #endif //__KEK__Stepper___INITIALIZATION__H__
```

A.2.7. Stepper Actuatuion.h

```
    #ifndef __KEK__Stepper___Actuation__H__
    #define __KEK__Stepper___Actuation__H__

    void Stepper_Motor_Actuate_Left(uint32_t* CCRs,float*
SteeringAngle)
    {
        float Kp = 1;    //Need a frequency range and corresponding
power settings

        GPIO_ResetBits(GPIOE, GPIO_Pin_0);    /*Motor 1 On/Off*/
```

```
        GPIO_ResetBits(GPIOD, GPIO_Pin_3);    /*Motor 1 Direction*/
        TIM11->ARR  = CCRs[0];                          /*Motor 1
ARR*/
        TIM11->CCR1 = CCRs[0]/2;                   /*Motor 1 PWM*/

        GPIO_ResetBits(GPIOE, GPIO_Pin_1);    /*Motor 2 On/Off*/
        GPIO_ResetBits(GPIOD, GPIO_Pin_2);    /*Motor 2 Direction*/
        TIM10->ARR  = CCRs[1];                          /*Motor 2
ARR*/
        TIM10->CCR1 = CCRs[1]/2;                   /*Motor 2 PWM*/

        GPIO_ResetBits(GPIOE, GPIO_Pin_2);    /*Motor 3 On/Off*/
        GPIO_ResetBits(GPIOD, GPIO_Pin_1);    /*Motor 3 Direction*/
        TIM12->ARR  = CCRs[2];                          /*Motor 3
ARR*/
        TIM12->CCR1 = CCRs[2]/2;                   /*Motor 3 PWM*/

        GPIO_ResetBits(GPIOE, GPIO_Pin_3);    /*Motor 4 On/Off*/
        GPIO_SetBits(GPIOD, GPIO_Pin_0); /*Motor 4 Direction*/
        TIM9->ARR  = CCRs[3];                      /*Motor 4 ARR*/
        TIM9->CCR2 = CCRs[3]/2;                         /*Motor 4
PWM*/
    }

    void Stepper_Motor_Actuate_Right(uint32_t* CCRs,float*
SteeringAngle)
    {
        float Kp = 1;   //Need a frequency range and corresponding
power settings

        GPIO_ResetBits(GPIOE, GPIO_Pin_0);    /*Motor 1 On/Off*/
        GPIO_SetBits(GPIOD, GPIO_Pin_3); /*Motor 1 Direction*/
        TIM11->ARR  = CCRs[0];                          /*Motor 1
ARR*/
        TIM11->CCR1 = CCRs[0]/2;                   /*Motor 1 PWM*/

        GPIO_ResetBits(GPIOE, GPIO_Pin_1);    /*Motor 2 On/Off*/
        GPIO_SetBits(GPIOD, GPIO_Pin_2); /*Motor 2 Direction*/
        TIM10->ARR  = CCRs[1];                          /*Motor 2
ARR*/
        TIM10->CCR1 = CCRs[1]/2;                   /*Motor 2 PWM*/

        GPIO_ResetBits(GPIOE, GPIO_Pin_2);    /*Motor 3 On/Off*/
        GPIO_SetBits(GPIOD, GPIO_Pin_1); /*Motor 3 Direction*/
        TIM12->ARR  = CCRs[2];                          /*Motor 3
ARR*/
        TIM12->CCR1 = CCRs[2]/2;                   /*Motor 3 PWM*/

        GPIO_ResetBits(GPIOE, GPIO_Pin_3);    /*Motor 4 On/Off*/
        GPIO_ResetBits(GPIOD, GPIO_Pin_0);    /*Motor 4 Direction*/
        TIM9->ARR  = CCRs[3];                      /*Motor 4 ARR*/
```

```
        TIM9->CCR2 = CCRs[3]/2;                          /*Motor 4
PWM*/
    }
    #endif //__KEK__Stepper___Actuation__H__
```

A.2.8. Stepper_Motor_Actuation.h

```
    #ifndef __KEK__Stepper__Motor___Actuation__H__
    #define __KEK__Stepper__Motor___Actuation__H__

    int sac1;
    int sac2;
    int sac3;
    int sac4;

    void Stepper_Motor_Action_Determination(float* SteeringAngle,
float* SteeringSetpoint, uint32_t* StepperAction){
        int i;
        float tol = 6; //degrees


        /*for(i=0; i<4; i++){
            if(abs(SteeringAngle[i]-SteeringSetpoint[i])>tol){
                if(SteeringAngle[i]>SteeringSetpoint[i]){
                    StepperAction[i] = 0;
                } else{
                    StepperAction[i] = 2;
                }
            }
            if(abs(SteeringAngle[i]-SteeringSetpoint[i])<tol){
                StepperAction[i] = 1;
            }
        */

        if(abs((SteeringAngle[0]-SteeringSetpoint[0]))>tol){
            if(SteeringAngle[0]>SteeringSetpoint[0]){
                StepperAction[0] = 0;
            } else{
                StepperAction[0] = 2;
            }
        }else{
            StepperAction[0] = 1;
        }
        if(abs(SteeringAngle[1]-SteeringSetpoint[1])>tol){
            if(SteeringAngle[1]>SteeringSetpoint[1]){
                StepperAction[1] = 0;
            } else{
                StepperAction[1] = 2;
            }
        }else{
            StepperAction[1] = 1;
```

```
		}
		//if(abs(SteeringAngle[2]-SteeringSetpoint[2])>tol){
		//	if(SteeringAngle[2]>SteeringSetpoint[2]){
		//		StepperAction[2] = 0;
		//	} else{
		//		StepperAction[2] = 2;
		//	}
		//	}else{
		//	StepperAction[2] = 1;
		//}
		if(abs(SteeringAngle[3]-SteeringSetpoint[3])>tol){
			if(SteeringAngle[3]>SteeringSetpoint[3]){
				StepperAction[3] = 0;
			} else{
				StepperAction[3] = 2;
			}
		}else{
			StepperAction[3] = 1;
		}
		sac1 = StepperAction[0];
		sac2 = StepperAction[1];
		sac3 = StepperAction[2];
		sac4 = StepperAction[3];
	}

	void Stepper_Motor_Actuate(uint32_t* CCRs,uint32_t*
StepperAction)
	{
		int i;
		float Kp = 1;	//Need a frequency range and corresponding
power settings

		sac1 = StepperAction[0];
		sac2 = StepperAction[1];
		sac3 = StepperAction[2];
		sac4 = StepperAction[3];

		if(sac1 == 0){
			GPIO_ResetBits(GPIOE, GPIO_Pin_0);	/*Motor 1
On/Off*/
			GPIO_ResetBits(GPIOD, GPIO_Pin_3);	/*Motor 1
Direction*/
			TIM11->ARR  = CCRs[1];			/*Motor 1
ARR*/
			TIM11->CCR1 = CCRs[1]/2;		/*Motor 1 PWM*/
			}
		if(sac1 == 1){
			GPIO_SetBits(GPIOE, GPIO_Pin_0); /*Motor 1 On/Off*/
			GPIO_ResetBits(GPIOD, GPIO_Pin_3);	/*Motor 1
Direction*/
			TIM11->ARR  = 0;		/*Motor 1 ARR*/
			TIM11->CCR1 = 0;		/*Motor 1 PWM*/
```

```
            }
        if(sac1 == 2){
            GPIO_ResetBits(GPIOE, GPIO_Pin_0);    /*Motor 1
On/Off*/
            GPIO_SetBits(GPIOD, GPIO_Pin_3); /*Motor 1 Direction*/
            TIM11->ARR  = CCRs[1];                       /*Motor 1
ARR*/
            TIM11->CCR1 = CCRs[1]/2;             /*Motor 1 PWM*/
            }
        if(sac2 == 0){
            GPIO_ResetBits(GPIOE, GPIO_Pin_1);    /*Motor 2
On/Off*/
            GPIO_ResetBits(GPIOD, GPIO_Pin_2);    /*Motor 2
Direction*/
            TIM10->ARR  = CCRs[0];                       /*Motor 2
ARR*/
            TIM10->CCR1 = CCRs[0]/2;             /*Motor 2 PWM*/
            }
        if(sac2 == 1){
            GPIO_SetBits(GPIOE, GPIO_Pin_1); /*Motor 2 On/Off*/
            GPIO_SetBits(GPIOD, GPIO_Pin_2); /*Motor 2 Direction*/
            TIM10->ARR  = 0;                /*Motor 2 ARR*/
            TIM10->CCR1 = 0;           /*Motor 2 PWM*/
            }
        if(sac2 == 2){
            GPIO_ResetBits(GPIOE, GPIO_Pin_1);    /*Motor 2
On/Off*/
            GPIO_SetBits(GPIOD, GPIO_Pin_2); /*Motor 2 Direction*/
            TIM10->ARR  = CCRs[0];                       /*Motor 2
ARR*/
            TIM10->CCR1 = CCRs[0]/2;             /*Motor 2 PWM*/
            }
        if(sac3 == 0){
            GPIO_ResetBits(GPIOE, GPIO_Pin_2);    /*Motor 3
On/Off*/
            GPIO_ResetBits(GPIOD, GPIO_Pin_1);    /*Motor 3
Direction*/
            TIM12->ARR  = CCRs[2];                       /*Motor 3
ARR*/
            TIM12->CCR1 = CCRs[2]/2;             /*Motor 3 PWM*/
            }
        if(sac3 == 1){
            GPIO_SetBits(GPIOE, GPIO_Pin_2); /*Motor 3 On/Off*/
            GPIO_ResetBits(GPIOD, GPIO_Pin_1);    /*Motor 3
Direction*/
            TIM12->ARR  = 0;                /*Motor 3 ARR*/
            TIM12->CCR1 = 0;           /*Motor 3 PWM*/
            }
        if(sac3 == 2){
            GPIO_ResetBits(GPIOE, GPIO_Pin_2);    /*Motor 3
On/Off*/
            GPIO_SetBits(GPIOD, GPIO_Pin_1); /*Motor 3 Direction*/
```

```
                TIM12->ARR  = CCRs[2];                      /*Motor 3
ARR*/
                TIM12->CCR1 = CCRs[2]/2;              /*Motor 3 PWM*/
                }
           if(sac4 == 0){
                GPIO_ResetBits(GPIOE, GPIO_Pin_3);    /*Motor 4
On/Off*/
                GPIO_SetBits(GPIOD, GPIO_Pin_0); /*Motor 4 Direction*/
                TIM9->ARR  = CCRs[3];                 /*Motor 4 ARR*/
                TIM9->CCR2 = CCRs[3]/2;                     /*Motor 4
PWM*/
                }
           if(sac4 == 1){
                GPIO_SetBits(GPIOE, GPIO_Pin_3); /*Motor 4 On/Off*/
                GPIO_SetBits(GPIOD, GPIO_Pin_0); /*Motor 4 Direction*/
                TIM9->ARR  = 0;                  /*Motor 4 ARR*/
                TIM9->CCR2 = 0;                  /*Motor 4 PWM*/
                }
           if(sac4 == 2){
                GPIO_ResetBits(GPIOE, GPIO_Pin_3);    /*Motor 4
On/Off*/
                GPIO_ResetBits(GPIOD, GPIO_Pin_0);    /*Motor 4
Direction*/
                TIM9->ARR  = CCRs[3];                 /*Motor 4 ARR*/
                TIM9->CCR2 = CCRs[3]/2;                     /*Motor 4
PWM*/
                }

      }
      /*
      void Stepper_Motor_Actuate_Right(uint32_t* CCRs,float*
SteeringAngle)
      {
           float Kp = 1;   //Need a frequency range and corresponding
power settings

           GPIO_ResetBits(GPIOE, GPIO_Pin_0);    /*Motor 1 On/Off*/
           //GPIO_SetBits(GPIOD, GPIO_Pin_3);    /*Motor 1 Direction*/
           //TIM11->ARR  = CCRs[0];                     /*Motor 1
ARR*/
           //TIM11->CCR1 = CCRs[0]/2;              /*Motor 1 PWM*/

           //GPIO_ResetBits(GPIOE, GPIO_Pin_1);  /*Motor 2 On/Off*/
           //GPIO_SetBits(GPIOD, GPIO_Pin_2);    /*Motor 2 Direction*/
           //TIM10->ARR  = CCRs[1];                     /*Motor 2
ARR*/
           //TIM10->CCR1 = CCRs[1]/2;              /*Motor 2 PWM*/

           //GPIO_ResetBits(GPIOE, GPIO_Pin_2);  /*Motor 3 On/Off*/
           //GPIO_SetBits(GPIOD, GPIO_Pin_1);    /*Motor 3 Direction*/
           //TIM12->ARR  = CCRs[2];                     /*Motor 3
ARR*/
```

```
            //TIM12->CCR1 = CCRs[2]/2;                /*Motor 3 PWM*/

            //GPIO_ResetBits(GPIOE, GPIO_Pin_3);  /*Motor 4 On/Off*/
            //GPIO_ResetBits(GPIOD, GPIO_Pin_0);  /*Motor 4 Direction*/
            //TIM9->ARR  = CCRs[3];                          /*Motor 4
ARR*/
            //TIM9->CCR2 = CCRs[3]/2;                        /*Motor 4
PWM*/
      //}
      //*/
      #endif
```

A.2.9. ADC_Initialization.h

```
      #ifndef __KARL__GPIO__Potentiometer____Initializations__H__
      #define __KARL__GPIO__Potentiometer____Initializations__H__

      #include <stm32f4xx_rcc.h>
      #include <stm32f4xx_gpio.h>
      #include <stm32f4xx_tim.h>
      #include "stm32f4xx_dma.h"
      #include "stm32f4xx_adc.h"

      void RCC_Configuration(void);
      void GPIO__Potentiometer__Initializations(void);
      void ADC_Configuration(void);
      void DMA_Configuration(uint16_t* memBuffer);

      /*****************************************************************
**********************/

      void RCC_Configuration(void) {
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC |
RCC_AHB1Periph_GPIOA | RCC_AHB1Periph_GPIOB | RCC_AHB1Periph_DMA2,
ENABLE);
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
      }

      /*****************************************************************
**********************/
      void GPIO__Potentiometer__Initializations(void) {
            GPIO_InitTypeDef GPIO_InitStructure;  //

            /*
             * Need to use either ADC1 or ADC2, ADC3 is not connected
to all analog pins
             *
             * PB0 = Analog Channel 8
             * PB1 = Analog Channel 9
             * PC4 = Analog Channel 14
             * PA3 = Analog Channel 3
```

```
          */

          // Port B pin init
          GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1;
          GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN;
          GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL ;
          GPIO_Init(GPIOB, &GPIO_InitStructure);

          // Port C pin init
         GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;
         GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN;
         GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL ;
         GPIO_Init(GPIOC, &GPIO_InitStructure);

         // Port A pin init
         GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
         GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN;
         GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL ;
         GPIO_Init(GPIOA, &GPIO_InitStructure);
     }

     /***********************************************************
**********************/

     void ADC_Configuration(void) {

         ADC_CommonInitTypeDef   ADC_CommonInitStructure; //
         ADC_InitTypeDef         ADC_InitStructure;   //

         /* ADC Common Init */
         ADC_CommonInitStructure.ADC_Mode             =
ADC_Mode_Independent;
         ADC_CommonInitStructure.ADC_Prescaler        =
ADC_Prescaler_Div2;
         ADC_CommonInitStructure.ADC_DMAAccessMode    =
ADC_DMAAccessMode_Disabled;
         ADC_CommonInitStructure.ADC_TwoSamplingDelay =
ADC_TwoSamplingDelay_5Cycles;
         ADC_CommonInit(&ADC_CommonInitStructure);

         ADC_InitStructure.ADC_Resolution           =
ADC_Resolution_12b;
         ADC_InitStructure.ADC_ScanConvMode         = ENABLE; //
multiple channel
         ADC_InitStructure.ADC_ContinuousConvMode   = ENABLE; //
Conversions Triggered, disable to manually do ADC ops
         ADC_InitStructure.ADC_ExternalTrigConvEdge =
ADC_ExternalTrigConvEdge_None; // Manual
         ADC_InitStructure.ADC_ExternalTrigConv     = 0;
         ADC_InitStructure.ADC_DataAlign            =
ADC_DataAlign_Right;
```

```
        ADC_InitStructure.ADC_NbrOfConversion      = 4; // want to
read analog values from 4 pins
        ADC_Init(ADC1, &ADC_InitStructure);

        /* ADC1 channel(s) configuration */
        ADC_RegularChannelConfig(ADC1, ADC_Channel_8,  1,
ADC_SampleTime_144Cycles); // PB0, front left
        ADC_RegularChannelConfig(ADC1, ADC_Channel_9,  2,
ADC_SampleTime_144Cycles); // PB1, front right
        ADC_RegularChannelConfig(ADC1, ADC_Channel_14, 3,
ADC_SampleTime_144Cycles); // PC4, rear left
        ADC_RegularChannelConfig(ADC1, ADC_Channel_3,  4,
ADC_SampleTime_144Cycles); // PA3, rear right

    }

    /**************************************************************
*********************/

    void DMA_Configuration(uint16_t* memBuffer) {
         /*
          * DMA (Direct Memory Access) is used to automatically
          * transfer multiple analog reads to a known location in
memory
          */

         DMA_InitTypeDef DMA_InitStruct;  //

         DMA_InitStruct.DMA_Channel            = 2;
         DMA_InitStruct.DMA_PeripheralBaseAddr = (uint32_t)&ADC1-
>DR;            // ADC1's data register
         DMA_InitStruct.DMA_Memory0BaseAddr    =
(uint32_t)memBuffer;            // actual location to put data, passed
as an argument
        DMA_InitStruct.DMA_DIR                =
DMA_DIR_PeripheralToMemory;      // direction memory travels during
DMA operation
        DMA_InitStruct.DMA_BufferSize         = 4;
// number of (32-bit) values to copy
        DMA_InitStruct.DMA_PeripheralInc      =
DMA_PeripheralInc_Disable;       // ??
        DMA_InitStruct.DMA_MemoryInc          = DMA_MemoryInc_Enable;
// ??
        DMA_InitStruct.DMA_PeripheralDataSize =
DMA_PeripheralDataSize_HalfWord; // because peripheral produces 16-bit
values
        DMA_InitStruct.DMA_MemoryDataSize     =
DMA_MemoryDataSize_HalfWord;     // because buffer stored 16-bit
values

        // bunch of stuff pertaining to actual DMA transfer
        DMA_InitStruct.DMA_Mode            = DMA_Mode_Circular;
```

```
        DMA_InitStruct.DMA_Priority        = DMA_Priority_High;
        DMA_InitStruct.DMA_FIFOMode        = DMA_FIFOMode_Disable;
        DMA_InitStruct.DMA_FIFOThreshold   =
DMA_FIFOThreshold_HalfFull;
        DMA_InitStruct.DMA_MemoryBurst     = DMA_MemoryBurst_Single;
        DMA_InitStruct.DMA_PeripheralBurst =
DMA_PeripheralBurst_Single;

        DMA_Init(DMA2_Stream0, &DMA_InitStruct);
        DMA_Cmd(DMA2_Stream0, ENABLE);
    }

    /***********************************************************
*********************/

    #endif // __KARL__GPIO__Potentiometer____Initializations__H__
```

A.2.10. ADC_Measurement.h

```
    #ifndef __KARL__POTENTIOMETER____Position__NEW__H__
    #define __KARL__POTENTIOMETER____Position__NEW__H__
    // STM32 ADC1 CH11 (PC.1) STM32F4 Discovery - sourcer32@gmail.com

    #include "stm32f4xx.h"
    //#include "stm32f4_discovery.h"
    #include "stm32f4xx_gpio.h"
    #include "stm32f4xx_rcc.h"


    //#include "stm32f4xx_usart.h"
    //#include "stm32f4xx_delay.h"

    #define DEBUGGING_ADC

    float S1;
    float S2;
    float S3;
    float S4;

    uint16_t SS1;
    uint16_t SS3;
    uint16_t SS2;
    uint16_t SS4;


    uint16_t motorValues[4];

    #define MOTOR_FRONT_LEFT  0
    #define MOTOR_FRONT_RIGHT 1
    #define MOTOR_BACK_LEFT   2
    #define MOTOR_BACK_RIGHT  3
```

```
#define MOTOR_CONTROL_THRESHOLD 2048 // out of 4095 or ~50%

void SteeringMotorAngles(float* SteeringAngle)
{

          SS1 = motorValues[MOTOR_FRONT_LEFT];
          SS2 = motorValues[MOTOR_FRONT_RIGHT];
          SS3 = motorValues[MOTOR_BACK_LEFT];
          SS4 = motorValues[MOTOR_BACK_RIGHT];
            //slope                                x
intercept
     S1=.1197*(SS1-2092);
          S2=.0807*(SS2-3038);
     S3=.0776*(SS3-1884);
     S4=.0825*(SS4-2409);

     //if(OffsetArray != 0){
          //    int i;
          //    S1 -= OffsetArray[0];
          //    S2 -= OffsetArray[1];
          //    S3 -= OffsetArray[2];
     //    S4 -= OffsetArray[3];

          //}

     SteeringAngle[0] = S1;
     SteeringAngle[1] = S2;
     SteeringAngle[2] = S3;
     SteeringAngle[3] = S4;



}

#endif //__KARL__POTENTIOMETER____Position__NEW__H__
```

A.2.11. ENC_Initialization.h

```
#ifndef __KARL__ENCODER____Initializations__H__
#define __KARL__ENCODER____Initializations__H__

#include <stm32f4xx_rcc.h>
#include <stm32f4xx_gpio.h>
#include <stm32f4xx_tim.h>

void GPIO__Encoder__and_TIM__Initializations(void)
{
     GPIO_InitTypeDef  GPIO_InitStructure;
```

```
    //%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        //Encoder Initializations

        //ENCODER 1A->PE9 -> TIM1-CH1
        //ENCODER 1B->PE11-> TIM1-CH2

        //ENCODER 2A->PC6 -> TIM8-CH1
        //ENCODER 2B->PC7 -> TIM8-CH2

        //ENCODER 3A->PA6 -> TIM5-CH1
        //ENCODER 3B->PA7 -> TIM5-CH2

        //ENCODER 4A->PA1 -> TIM3-CH1
        //ENCODER 4B->PA0 -> TIM3-CH2

       //  Initialize the peripheral clocks.
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOE, ENABLE);  //ENC
1
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);  //ENC
2
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);  //ENC
3 && ENC 4

        RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM1, ENABLE);   //ENC
1
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM8, ENABLE);   //ENC
2
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM5, ENABLE);   //ENC
3
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);   //ENC
4

        //Enable Encoder 1 Pins
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9|GPIO_Pin_11;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
        GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
        GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
        GPIO_Init(GPIOE, &GPIO_InitStructure);

        //Enable Encoder 2 Pins
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6|GPIO_Pin_7;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
        GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
        GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
        GPIO_Init(GPIOC, &GPIO_InitStructure);

        //Enable Encoder 3 Pins
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
```

```
            GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
            GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
            GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
            GPIO_Init(GPIOA, &GPIO_InitStructure);


            //Enable Encoder 4 Pins
            GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6|GPIO_Pin_7;
            GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
            GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
            GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
            GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
            GPIO_Init(GPIOA, &GPIO_InitStructure);

            //Connect TIM Encoder pins to GPIO
            GPIO_PinAFConfig(GPIOE, GPIO_PinSource9,  GPIO_AF_TIM1);
// TIM1 CH.1->ENC 1A
            GPIO_PinAFConfig(GPIOE, GPIO_PinSource11, GPIO_AF_TIM1);
// TIM1 CH.2->ENC 1B

            GPIO_PinAFConfig(GPIOC, GPIO_PinSource6,  GPIO_AF_TIM8);
// TIM8 CH.1->ENC 2A
            GPIO_PinAFConfig(GPIOC, GPIO_PinSource7,  GPIO_AF_TIM8);
// TIM8 CH.2->ENC 2B

            GPIO_PinAFConfig(GPIOA, GPIO_PinSource0,  GPIO_AF_TIM5);
// TIM5 CH.1->ENC 3A
            GPIO_PinAFConfig(GPIOA, GPIO_PinSource1,  GPIO_AF_TIM5);
// TIM5 CH.2->ENC 3B

            GPIO_PinAFConfig(GPIOA, GPIO_PinSource6,  GPIO_AF_TIM3);
// TIM3 CH.1->ENC 4A
            GPIO_PinAFConfig(GPIOA, GPIO_PinSource7,  GPIO_AF_TIM3);
// TIM3 CH.2->ENC 4B

            TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
            TIM_ICInitTypeDef TIM_ICInitStruct;

            TIM_TimeBaseStructure.TIM_Prescaler = 0;
            TIM_TimeBaseStructure.TIM_Period = 0xFFFFFFFF; // Maximal
            TIM_TimeBaseStructure.TIM_ClockDivision = 0;
            TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
            TIM_TimeBaseInit(TIM1, &TIM_TimeBaseStructure);  //ENC 1
            TIM_TimeBaseInit(TIM8, &TIM_TimeBaseStructure);  //ENC 2
            TIM_TimeBaseInit(TIM5, &TIM_TimeBaseStructure);  //ENC 3
            TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);  //ENC 4
            TIM_EncoderInterfaceConfig(TIM1,TIM_EncoderMode_TI1,TIM_ICP
olarity_Rising,TIM_ICPolarity_Rising);//ENC 1
            TIM_EncoderInterfaceConfig(TIM8,TIM_EncoderMode_TI1,TIM_ICP
olarity_Rising,TIM_ICPolarity_Rising);//ENC 2
            TIM_EncoderInterfaceConfig(TIM5,TIM_EncoderMode_TI1,TIM_ICP
olarity_Rising,TIM_ICPolarity_Rising);//ENC 3
```

```
            TIM_EncoderInterfaceConfig(TIM3,TIM_EncoderMode_TI1,TIM_ICP
olarity_Rising,TIM_ICPolarity_Rising);//ENC 4

            TIM_ICInitStruct.TIM_Channel = TIM_Channel_1;
            TIM_ICInitStruct.TIM_ICPolarity = TIM_ICPolarity_Rising;
            TIM_ICInitStruct.TIM_ICSelection =
TIM_ICSelection_DirectTI;
            TIM_ICInitStruct.TIM_ICPrescaler = TIM_ICPSC_DIV1;
            TIM_ICInitStruct.TIM_ICFilter = 0xFF;
            TIM_ICInit(TIM1,&TIM_ICInitStruct);    //ENC 1
            TIM_ICInit(TIM8,&TIM_ICInitStruct);    //ENC 2
            TIM_ICInit(TIM5,&TIM_ICInitStruct);    //ENC 3
            TIM_ICInit(TIM3,&TIM_ICInitStruct);    //ENC 4
            TIM_ICInitStruct.TIM_Channel = TIM_Channel_2;
            TIM_ICInit(TIM1,&TIM_ICInitStruct);    //ENC 1
            TIM_ICInit(TIM8,&TIM_ICInitStruct);    //ENC 2
            TIM_ICInit(TIM5,&TIM_ICInitStruct);    //ENC 3
            TIM_ICInit(TIM3,&TIM_ICInitStruct);    //ENC 4
            TIM_Cmd(TIM1, ENABLE);      //ENC 1
            TIM_Cmd(TIM8, ENABLE);      //ENC 2
            TIM_Cmd(TIM5, ENABLE);      //ENC 5
            TIM_Cmd(TIM3, ENABLE);      //ENC 3
      }

      #endif     //__KARL__ENCODER____Initializations__H__
```

A.2.12. ENC_measurement.h

```
      #ifndef __KARL__ENCODER____Measurement__H__
      #define __KARL__ENCODER____Measurement__H__

      #define Velocity_1 0x01
      #define Velocity_2 0x02
      #define Velocity_3 0x03
      #define Velocity_4 0x04

      float V1;
      float V2;
      float V3;
      float V4;

      float VS1;
      float VS2;
      float VS3;
      float VS4;

      int G11;
      int G21;
      int G31;
      int G41;
```

```
      int G12;
      int G22;
      int G32;
      int G42;

      float  EV1;
      float  EV2;
      float  EV3;
      float  EV4;

      //int T1;
      int T3;
      int True = 0;


      /*This is a function to reset the TIM count from the encoders.
Eventually the encoders reach a maximum value and the velocity
measurements stop. This should prevent that.*/
      void TIM_ResetCounter(TIM_TypeDef* TIMx)
      {
        /* Check the parameters */
        assert_param(IS_TIM_ALL_PERIPH(TIMx));

        /* Reset the Counter Register value */
        TIMx->CNT = 0;
      }

      void Get_Wheel_Velocity(float* WheelVelocity, float*
VelocitySetpoint, float* VelocityError) {
            int i = 0;
            int  Flag[4] = {0,0,0,0};
            int InitialTime = TM_Time;
            int ticks = 40;

            VS1 = VelocitySetpoint[0];
            VS2 = VelocitySetpoint[1];
            VS3 = VelocitySetpoint[2];
            VS4 = VelocitySetpoint[3];

            True = 0;

            if(G12>50000){
                  TIM_ResetCounter(TIM1);
            }

            if(G22>50000){
                  TIM_ResetCounter(TIM8);
            }

            if(G32<10000){
                  TIM_ResetCounter(TIM5);
            }
```

```
if(G42<10000){
      TIM_ResetCounter(TIM3);
}

G11=TIM_GetCounter(TIM1);
G21=TIM_GetCounter(TIM8);
G31=TIM_GetCounter(TIM5);
G41=TIM_GetCounter(TIM3);

while(True<1){

      G12=TIM_GetCounter(TIM1);
      G22=TIM_GetCounter(TIM8);
      G32=TIM_GetCounter(TIM5);
      G42=TIM_GetCounter(TIM3);

      if ((G12-G11)>ticks && Flag[0]==0){
            T3 = TM_Time;
            V1 = ticks*0.025512695*1000/(T3-InitialTime);
            WheelVelocity[0] = V1;
            EV1 = VS1 - V1;
            VelocityError[0] = EV1;
            Flag[0] = 1;
      }

      if ((G22-G21)>ticks && Flag[1]==0){
            T3 = TM_Time;
            V2 = ticks*0.025512695*1000/(T3-InitialTime);
            WheelVelocity[1] = V2;
            EV2 = VS2 - V2;
            VelocityError[1] = EV2;
            Flag[1] = 1;
      }

      if ((G31-G32)>ticks && Flag[2]==0){
            T3 = TM_Time;
            V3 = ticks*0.025512695*1000/(T3-InitialTime);
            WheelVelocity[2] = V3;
            EV3 = VS3 - V3;
            VelocityError[2] = EV3;
            Flag[2] = 1;
      }

      if ((G41-G42)>ticks && Flag[3]==0){
            T3 = TM_Time;
            V4 = ticks*0.025512695*1000/(T3-InitialTime);
            WheelVelocity[3] = V4;
            EV4 = VS4 - V4;
            VelocityError[3] = EV4;
            Flag[3] = 1;
      }
```

```
                if(Flag[0]==1 && Flag[1]==1&& Flag[2]==1 &&
Flag[3]==1){
                      True = 1;
                }

                if(i == 5000000){
                      True = 1;
                      if(Flag[0]<1){
                            V1 = 0;
                            WheelVelocity[0] = V1;
                            EV1 = VS1 - V1;
                            VelocityError[0] = EV1;
                      }
                      if(Flag[1]<1){
                            V2 = 0;
                            WheelVelocity[1] = V2;
                            EV2 = VS2 - V2;
                            VelocityError[1] = EV2;
                      }
                      if(Flag[2]<1){
                            V3 = 0;
                            WheelVelocity[2] = V3;
                            EV3 = VS3 - V3;
                            VelocityError[2] = EV3;
                      }
                      if(Flag[3]<1){
                            V4 = 0;
                            WheelVelocity[3] = V4;
                            EV4 = VS4 - V4;
                            VelocityError[3] = EV4;
                      }
                }
                i = i+1;
          }
    }

    #endif     //__KARL__ENCODER____Measurement__H__

    //if(Flag[0]==1 && Flag[1]==1 && Flag[3]==1){

    //http://www.robotc.net/wikiarchive/Tutorials/Arduino_Projects/Mo
bile_Robotics/VEX/Using_encoders_to_drive_straight
    //https://www.tetrixrobotics.com/GettingStartedGuide/files/addons
/encoders/Programming/programmingGuides/RC_ProgGuide.pdf
    //https://www.toptal.com/robotics/programming-a-robot-an-
introductory-tutorial
    //https://arduino.stackexchange.com/questions/24437/dc-motor-
speed-measurement-using-rotary-encoder


    /*
```

```
	if ((TIM_GetCounter(TIM1)-WheelVelocityInitialBits[0])>50){
		//TM_RTC_GetDateTimeFromUnix(&Time, TM_Time);
	   T2 = TM_Time;
		WheelVelocity[0] = 50*0.025512695*1000/(T2-T1);
	}

	if ((TIM_GetCounter(TIM8)-WheelVelocityInitialBits[1])>50){
		//TM_RTC_GetDateTimeFromUnix(&Time, TM_Time);
	   //T2 = Time.unix;
	  // T2 = TM_Time;
		T2 = TM_Time;
	   WheelVelocity[1] = 50*0.025512695*1000/(T2-T1);
	}
	if ((TIM_GetCounter(TIM3)-WheelVelocityInitialBits[2])>50){
		//TM_RTC_GetDateTimeFromUnix(&Time, TM_Time);
	   //T2 = Time.unix;
		T2 = TM_Time;
		WheelVelocity[2] = 50*0.025512695*1000/(T2-T1);
	}

	if ((TIM_GetCounter(TIM5)-WheelVelocityInitialBits[3])>50){
		//TM_RTC_GetDateTimeFromUnix(&Time, TM_Time);
	   //T2 = Time.unix;
		T2 = TM_Time;
		WheelVelocity[3] = 50*0.025512695*1000/(T2-T1);
	}

	*/



	//WheelVelocityFinalBits[0] = TIM_GetCounter(TIM1);
	//WheelVelocityFinalBits[1] = TIM_GetCounter(TIM8);
	//WheelVelocityFinalBits[2] = TIM_GetCounter(TIM3);
	//WheelVelocityFinalBits[3] = TIM_GetCounter(TIM5);

	/*
	for(i=0;i=3;i=i+1){
	if (WheelVelocityFinalBits[i]<WheelVelocityInitialBits[i]){
		Buffer[i]=(65535-
WheelVelocityInitialBits[i])+WheelVelocityFinalBits[i];
	}
	else{
		Buffer[i]=WheelVelocityFinalBits[i]-
WheelVelocityInitialBits[i];
	}
	WheelVelocity[i] = 0.025512695*Buffer[i]/.050;  //Wheel Velocity
= .025512695(inches/bit)*(bits travelled)/(time difference)
	}


	*/
```

A.2.13. IMU_Initialization.h

```
      #ifndef __KEK__IMU___INITIALIZATION__H__
      #define __KEK__IMU___INITIALIZATION__H__

      #define UM7_SLAVESELECT_PORT GPIOA
      #define UM7_SLAVESELECT_PIN  GPIO_Pin_4

      #include "tm_stm32f4_gpio.h"    // General Purpose Input/Output
      #include "tm_stm32f4_spi.h"     // Serial Peripheral Interface
      #include "tm_stm32f4_delay.h"   // Delay
      #include "tm_stm32f4_usart.h"   // USART Peripheral
      #include "tm_stm32f4_disco.h"   // center lights on stm board
      #include "tm_stm32f4_usb_vcp.h" // Virtual COM Port
      #include "cstm_um7_interface.h" // needs the above 2 #defines to
work properly

      // ==============================================
      // custom libraries

      //float rollRate;
      //float yawRate;
      //float tiltRate;

      //MESSAGE __msg_t;

      void IMU_Initializations(void){

              // TM_DISCO_LedInit();    // initialize center leds
(various debug utilities)
           //TM_DISCO_ButtonInit(); // initialize user button
                //TM_DELAY_Init();       // initialize delay
configuration (needed for um7 communication)
                TM_USB_VCP_Init();     // initialize virtual COM port
(USB serial)

           // Init UM7 on SPI 1
                TM_SPI_InitFull(
                          SPI1,                        // SPI 1
                          TM_SPI_PinsPack_2,           // MOSI: PB5,
MISO: PB4, SCK: PB3
                          SPI_BaudRatePrescaler_256, // default
baudrate is 45MHz
                          TM_SPI_Mode_0,               // clock
polarity low, data transmit on rising edge
                          SPI_Mode_Master,             // spi master
                          SPI_FirstBit_MSB             // transmit data
Most Significant Bit first
                );

                // Init Slave Select for UM7 IMU (NSS)
```

```
                TM_GPIO_Init(
                           GPIOA,                // GPIO Port A
                           GPIO_Pin_4,           // Pin 4
                           TM_GPIO_Mode_OUT,     // Mode: output
                           TM_GPIO_OType_PP,     // Mode: Push/Pull
                           TM_GPIO_PuPd_NOPULL, // No pull up/down
resistor
                           TM_GPIO_Speed_Fast   // ...fast
                );
      }
      #endif //__KEK__IMU___INITIALIZATION__H__
```

A.2.14. IMU_measurement.h

```
      #ifndef __KEK__IMU___MEASUREMENT__H__
      #define __KEK__IMU___MEASUREMENT__H__

      #include "tm_stm32f4_delay.h"

      float Iroll;
      float Ipitch;
      float Iyaw;
      float Mag;

      int IMU_Flag = 0;

      void Zero_IMU_Rate_Bias(void){
            um7_NSS_Low(); // select um7 device
            um7_sendCalibrationCommand(SPI1);
            Delayms(10);
            um7_NSS_High(); // deselect um7 device
      }

      void  Get_IMU_Data(float* IMU){
            IMU_Flag = 0;

            while(IMU_Flag<1) {
                        // loop events here
                        //static um7measurement r, p, y;
                        //r.id = 'r';
                        //p.id = 'p';
                        //y.id = 'y';


                        um7_NSS_Low(); // select um7 device
                              IMU[0] = um7_getRateAxis(SPI1, UM7_AXIS_X);
                              IMU[1] = um7_getRateAxis(SPI1, UM7_AXIS_Y);
                              IMU[2]  = um7_getRateAxis(SPI1,
UM7_AXIS_Z);
                              IMU[3] = um7_getAngle(SPI1, UM7_ANGLE_YAW);
                              Delayms(10);
```

```
                    um7_NSS_High(); // deselect um7 device

                    Iroll = IMU[0];
                    Ipitch = IMU[1];
                    Iyaw = IMU[2];
                    Mag = IMU[3];

                    //__msg_t.id = MSGID_um7RateGyroX;
                    //__msg_t.payload.f = rollRate;

          /*
                    um7_NSS_Low();
                         //y.data = um7_getAngle(SPI1,
UM7_ANGLE_YAW);
                         //y.data = um7_getAngle(SPI1,
UM7_ANGLE_ROLL);
                         //p.data = um7_getAngle(SPI1,
UM7_ANGLE_PITCH);
                         Delayms(1);
                    um7_NSS_High();
          */
          /*
                    // send the data over the serial link
                    if(TM_USB_VCP_GetStatus() ==
TM_USB_VCP_CONNECTED) {
                         TM_DISCO_LedOn(LED_GREEN);
                         TM_DISCO_LedOff(LED_RED);
                         //TM_USB_VCP_Send((uint8_t*)&r, 5); // send
roll
                         //TM_USB_VCP_Send((uint8_t*)&p, 5); // send
pitch
                         TM_USB_VCP_Send((uint8_t*)&y, 5); // send
yaw
                    } else {
                         TM_DISCO_LedOn(LED_RED);
                         TM_DISCO_LedOff(LED_GREEN);
                    }
          */
                         IMU_Flag=1;
                    }
     }
     #endif //__KEK__IMU___MEASUREMENT__H__
```

A.2.15. cstm_um7_interface

```
     #ifndef __JJC__UM7__INTERFACE__H__
     #define __JJC__UM7__INTERFACE__H__

     #include "tm_stm32f4_spi.h"
```

```
	// Slave Select port and pin needs to be defined outside of this
file
	#ifndef UM7_SLAVESELECT_PORT
	#error um7 Slave Select port (UM7_SLAVESELECT_PORT) needs to be
defined
	#endif // UM7_SLAVESELECT_PORT

	#ifndef UM7_SLAVESELECT_PIN
	#error um7 Slave Select pin (UM7_SLAVESELECT_PIN) needs to be
defined
	#endif // UM7_SLAVESELECT_PIN

	typedef enum {
		UM7_AXIS_X = 0, UM7_AXIS_Y = 1, UM7_AXIS_Z = 2
	} UM7_AXIS;

	typedef enum {
		UM7_ANGLE_ROLL = 0, UM7_ANGLE_PITCH = 1, UM7_ANGLE_YAW = 2
	} UM7_ANGLE;

	typedef struct {
		float roll;
		float pitch;
		float yaw;
	} um7_RollPitchYaw;

	// mostly used internally
	typedef enum {
		UM7_CMD_getRollPitch = 0,
		UM7_CMD_getYaw       = 1,
		UM7_CMD_getRateGyroX = 2,
		UM7_CMD_getRateGyroY = 3,
		UM7_CMD_getRateGyroZ = 4,
		UM7_CMD_ZeroGyros    = 5
	} UM7_CMD;

	// macros for setting/resetting slave select pin
	#define um7_NSS_High()    TM_GPIO_SetPinHigh(
UM7_SLAVESELECT_PORT, UM7_SLAVESELECT_PIN)
	#define um7_NSS_Low()     TM_GPIO_SetPinLow(
UM7_SLAVESELECT_PORT, UM7_SLAVESELECT_PIN)

	// return 1 on success, 0 on failure
	uint16_t um7_sendCommand(SPI_TypeDef* spix, UM7_CMD cmd) {
		uint8_t sendCmd[2] = {0x00, 0x00};
		switch(cmd) {
			case UM7_CMD_getRollPitch:
				sendCmd[1] = 0x70; break;
			case UM7_CMD_getYaw:
				sendCmd[1] = 0x71; break;
			case UM7_CMD_getRateGyroX:
				sendCmd[1] = 0x61; break;
```

```
			case UM7_CMD_getRateGyroY:
				sendCmd[1] = 0x62; break;
			case UM7_CMD_getRateGyroZ:
				sendCmd[1] = 0x63; break;
			case UM7_CMD_ZeroGyros:
				sendCmd[0] = 0x01;
				sendCmd[1] = 0xAD; break;
			default:
				return 0; // dont send anything over the SPI line
		}

		TM_SPI_Send(spix, sendCmd[0]);
		Delay(6); // data sheet says to wait 5 usec, we're playing
it safe
		TM_SPI_Send(spix, sendCmd[1]);
		Delay(6); // ...

		return 1;
	}

	uint16_t um7_sendCalibrationCommand(SPI_TypeDef* spix) {
		uint8_t sendCmd[6] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
				sendCmd[0] = 0x01;
				sendCmd[1] = 0xAD;
				sendCmd[2] = 0x00;
				sendCmd[3] = 0x00;
				sendCmd[4] = 0x00;
				sendCmd[5] = 0x00;

		TM_SPI_Send(spix, sendCmd[0]);
		Delay(6); // data sheet says to wait 5 usec, we're playing
it safe
		TM_SPI_Send(spix, sendCmd[1]);
		Delay(6); // ...
		TM_SPI_Send(spix, sendCmd[2]);
		Delay(6); // data sheet says to wait 5 usec, we're playing
it safe
		TM_SPI_Send(spix, sendCmd[3]);
		Delay(6); // ...
		TM_SPI_Send(spix, sendCmd[4]);
		Delay(6); // data sheet says to wait 5 usec, we're playing
it safe
		TM_SPI_Send(spix, sendCmd[5]);
		Delay(6); // ...

		return 1;
	}

	// buffer is expected to have 4 bytes of free space
	void um7_readReply4Byte(SPI_TypeDef* spix, uint8_t* buffer) {
		TM_SPI_ReadMulti(spix, buffer + 0, 0x00, 1);
		Delay(6);
```

```
	TM_SPI_ReadMulti(spix, buffer + 1, 0x00, 1);
	Delay(6);
	TM_SPI_ReadMulti(spix, buffer + 2, 0x00, 1);
	Delay(6);
	TM_SPI_ReadMulti(spix, buffer + 3, 0x00, 1);
	Delay(6);
}

int16_t um7_byteSwapInt16(int16_t input) {
	typedef union {
		int16_t iData;
		uint8_t bytes[2];
	} ByteData;

	ByteData bd;
	bd.iData = input;

	uint8_t temp = bd.bytes[0];
	bd.bytes[0] = bd.bytes[1];
	bd.bytes[1] = temp;

	return bd.iData;
}

float um7_byteSwapFloat(float input) {
	typedef union {
		float fData;
		uint8_t bytes[4];
	} ByteData;

	ByteData bd;
	bd.fData = input;

	uint8_t temp = bd.bytes[0];
	bd.bytes[0] = bd.bytes[3];
	bd.bytes[3] = temp;

	temp = bd.bytes[1];
	bd.bytes[1] = bd.bytes[2];
	bd.bytes[2] = temp;

	return bd.fData; // return the, now byte-swapped, float
}

float um7_getAngle(SPI_TypeDef* spix, UM7_ANGLE angle) {
	int16_t reply[2];

	switch(angle) {
		case UM7_ANGLE_ROLL:
			um7_sendCommand(spix, UM7_CMD_getRollPitch);
			um7_readReply4Byte(spix, reply);
			{
```

```
                        float roll =
(float)um7_byteSwapInt16(reply[0]);
                        return (roll / 91.02222);
                    }
                    break;
                case UM7_ANGLE_PITCH:
                    um7_sendCommand(spix, UM7_CMD_getRollPitch);
                    um7_readReply4Byte(spix, reply);
                    {
                        float pitch =
(float)um7_byteSwapInt16(reply[1]);
                        return (pitch / 91.02222);
                    }
                    break;
                case UM7_ANGLE_YAW:
                    um7_sendCommand(spix, UM7_CMD_getYaw);
                    um7_readReply4Byte(spix, reply);
                    {
                        float yaw =
(float)um7_byteSwapInt16(reply[0]);
                        return (yaw / 91.02222f);
                    }
                    break;
                default:
                    break;
            }
            return um7_byteSwapFloat((float)3.14159f);
        }

        float um7_getRateAxis(SPI_TypeDef* spix, UM7_AXIS axis) {
            uint8_t x_axis_cmd[2] = {0x00, 0x61};
            uint8_t y_axis_cmd[2] = {0x00, 0x62};
            uint8_t z_axis_cmd[2] = {0x00, 0x63};

            int hasData = 0;
            int i; // for use in for loops
            float returnData = 0.0f;

            switch(axis) {
                case UM7_AXIS_X:
                    for(i = 0; i < 2; i++) {
                        TM_SPI_Send(spix, x_axis_cmd[i]);
                        Delay(6);
                    }
                    hasData = 1;
                    break;
                case UM7_AXIS_Y:
                    for(i = 0; i < 2; i++) {
                        TM_SPI_Send(spix, y_axis_cmd[i]);
                        Delay(6);
                    }
                    hasData = 1;
```

```
                break;
            case UM7_AXIS_Z:
                for(i = 0; i < 2; i++) {
                    TM_SPI_Send(spix, z_axis_cmd[i]);
                    Delay(6);
                }
                hasData = 1;
                break;
            default:
                break;
        }

        if(hasData) {
            uint8_t imuData[4];
            for(i = 0; i < 4; i++) {
                TM_SPI_ReadMulti(spix, imuData+i, 0x00, 1);
                Delay(6);
            }

            // byte swap the data


            returnData = *(float*)imuData;
        }

        return um7_byteSwapFloat(returnData);
    }

    #endif // __JJC__UM7__INTERFACE__H__
```

A.2.16. tm_stm32f4_delay

```
    /**
     * @author  Tilen Majerle
     * @email   tilen@majerle.eu
     * @website http://stm32f4-discovery.com
     * @link    http://stm32f4-discovery.com/2014/04/library-03-
stm32f429-discovery-system-clock-and-pretty-precise-delay-library/
     * @version v2.4
     * @ide     Keil uVision
     * @license GNU GPL v3
     * @brief   Pretty accurate delay functions with SysTick or any
other timer
     *
    @verbatim
       ----------------------------------------------------------------------
--------
         Copyright (C) Tilen Majerle, 2015

         This program is free software: you can redistribute it and/or
modify
```

```
         it under the terms of the GNU General Public License as
published by
         the Free Software Foundation, either version 3 of the
License, or
         any later version.

         This program is distributed in the hope that it will be
useful,
         but WITHOUT ANY WARRANTY; without even the implied warranty
of
         MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
         GNU General Public License for more details.

         You should have received a copy of the GNU General Public
License
         along with this program.  If not, see
<http://www.gnu.org/licenses/>.
        --------------------------------------------------------------
--------
     @endverbatim
      */
     #ifndef TM_DELAY_H
     #define TM_DELAY_H 240

     /* C++ detection */
     #ifdef __cplusplus
     extern "C" {
     #endif

     /**
      * @addtogroup TM_STM32F4xx_Libraries
      * @{
      */

     /**
      * @defgroup TM_DELAY
      * @brief    Pretty accurate delay functions with SysTick or any
other timer - http://stm32f4-discovery.com/2014/04/library-03-
stm32f429-discovery-system-clock-and-pretty-precise-delay-library/
      * @{
      *
     @verbatim
     If you are using GCC compiler, then your microseconds delay is
probably totally inaccurate.
     USE TIMER FOR DELAY, otherwise your delay will not be accurate.

     Another way is to use ARM compiler.
     @endverbatim
      *
      * As of version 2.0 you have now two possible ways to make a
delay.
```

```
     * The first (and default) is Systick timer. It makes interrupts
every 1ms.
     * If you want delay in "us" accuracy, there is simple pooling
(variable) mode.
     *
     *
     * The second (better) options is to use one of timers on F4xx
MCU.
     * Timer also makes an interrupts every 1ms (for count time)
instead of 1us as it was before.
     * For "us" delay, timer's counter is used to count ticks. It
makes a new tick each "us".
     * Not all MCUs have all possible timers, so this lib has been
designed that you select your own.
     *
     * \par Select custom TIM for delay functions
     *
     * By default, Systick timer is used for delay. If you want your
custom timer,
     * open defines.h file, add lines below and edit for your needs.
     *
    \code{.c}
    //Select custom timer for delay, here is TIM2 selected.
    //If you want custom TIMx, just replace number "2" for your TIM's
number.
    #define TM_DELAY_TIM                 TIM2
    #define TM_DELAY_TIM_IRQ             TIM2_IRQn
    #define TM_DELAY_TIM_IRQ_HANDLER TIM2_IRQHandler
    \endcode
     *
     *
     * With this setting (using custom timer) you have better
accuracy in "us" delay.
     * Also, you have to know, that if you want to use timer for
delay, you have to include additional files:
     *
     *    - CMSIS:
     *         - STM32F4xx TIM
     *         - MISC
     *    - TM:
     *         TM TIMER PROPERTIES
     *
     * Delay functions (Delay, Delayms) are now Inline functions.
     * This allows faster execution and more accurate delay.
     *
     * If you are working with Keil uVision and you are using Systick
for delay,
     * then set KEIL_IDE define in options for project:
     *    - Open "Options for target"
     *    - Tab "C/C++"
     *    - Under "Define" add "KEIL_IDE", without quotes
     *
```

```
       * \par Custom timers
       *
       * Custom timers are a way to make some tasks in a periodic
value.
       * As of version 2.4, delay library allows you to create custom
timer which count DOWN and when it reaches zero, callback is called.
       *
       * You can use variable settings for count, reload value and auto
reload feature.
       *
       * \par Changelog
       *
      @verbatim
       Version 2.4
        - May 26, 2015
        - Added support for custom timers which can be called
periodically

       Version 2.3
        - April 18, 2015
        - Fixed support for internal RC clock

       Version 2.2
        - January 12, 2015
        - Added support for custom function call each time 1ms
interrupt happen
        - Function is called TM_DELAY_1msHandler(void), with __weak
parameter
        - attributes.h file needed

       Version 2.1
        - GCC compiler fixes
        - Still prefer that you use TIM for delay if you are working
with ARM-GCC compiler

       Version 2.0
        - November 28, 2014
        - Delay library has been totally rewritten. Because Systick is
designed to be used
             in RTOS, it is not compatible to use it at the 2 places at
the same time.
             For that purpose, library has been rewritten.
        - Read full documentation above

       Version 1.0
        - First release
      @endverbatim
       *
       * \par Dependencies
       *
      @verbatim
       - STM32F4xx
```

```
       - STM32F4xx RCC:       Only if you want to use TIMx for delay
instead of Systick
       - STM32F4xx TIM:       Only if you want to use TIMx for delay
instead of Systick
       - MISC
       - defines.h
       - TM TIMER PROPERTIES: Only if you want to use TIMx for delay
instead of Systick
       - attribute.h
      @endverbatim
       */
      #include "stm32f4xx.h"
      #include "stm32f4xx_rcc.h"
      #include "defines.h"
      #include "attributes.h"
      /* If user selectable timer is selected for delay */
      #if defined(TM_DELAY_TIM)
      #include "misc.h"
      #include "stm32f4xx_tim.h"
      #include "tm_stm32f4_timer_properties.h"
      #endif
      #include "stdlib.h"

      /**
       * @defgroup TM_DELAY_Typedefs
       * @brief    Library Typedefs
       * @{
       */

      /**
       * @brief  Custom timer structure
       */
      typedef struct {
           uint32_t ARR;             /*!< Auto reload value */
           uint32_t AutoReload;      /*!< Set to 1 if timer should be
auto reloaded when it reaches zero */
           uint32_t CNT;             /*!< Counter value, counter
counts down */
           uint8_t Enabled;          /*!< Set to 1 when timer is
enabled */
           void (*Callback)(void *); /*!< Callback which will be
called when timer reaches zero */
           void* UserParameters;     /*!< Pointer to user parameters
used for callback function */
      } TM_DELAY_Timer_t;

      /**
       * @}
       */

      /**
       * @defgroup TM_DELAY_Macros
```

```
 * @brief    Library Macros
 * @{
 */

/**
 * @brief  Number of allowed custom timers
 * @note   Should be changes in defines.h file if necessary
 */
#ifndef DELAY_MAX_CUSTOM_TIMERS
#define DELAY_MAX_CUSTOM_TIMERS   5
#endif

/* Memory allocation function */
#ifndef LIB_ALLOC_FUNC
#define LIB_ALLOC_FUNC    malloc
#endif

/* Memory free function */
#ifndef LIB_FREE_FUNC
#define LIB_FREE_FUNC     free
#endif

/**
 * @}
 */

/**
 * @defgroup TM_DELAY_Variables
 * @brief    Library Variables
 * @{
 */

/**
 * This variable can be used in main
 * It is automatically increased every time systick make an
interrupt
 */
extern __IO uint32_t TM_Time;
extern __IO uint32_t TM_Time2;
extern __IO uint32_t mult;

/**
 * @}
 */

/**
 * @defgroup TM_DELAY_Functions
 * @brief    Library Functions
 * @{
 */

/**
```

```
 * @param  Delays for specific amount of microseconds
 * @param  micros: Time in microseconds for delay
 * @retval None
 * @note   Declared as static inline
 */
static __INLINE void Delay(uint32_t micros) {
#if defined(TM_DELAY_TIM)
     volatile uint32_t timer = TM_DELAY_TIM->CNT;

     do {
          /* Count timer ticks */
          while ((TM_DELAY_TIM->CNT - timer) == 0);

          /* Increase timer */
          timer = TM_DELAY_TIM->CNT;

          /* Decrease microseconds */
     } while (--micros);
#else
     uint32_t amicros;

     /* Multiply micro seconds */
     amicros = (micros) * (mult);

     #ifdef __GNUC__
          if (SystemCoreClock == 180000000 || SystemCoreClock ==
100000000) {
               amicros -= mult;
          }
     #endif

     /* If clock is 100MHz, then add additional multiplier */
     /* 100/3 = 33.3 = 33 and delay wouldn't be so accurate */
     #if defined(STM32F411xE)
     amicros += mult;
     #endif

     /* While loop */
     while (amicros--);
#endif /* TM_DELAY_TIM */
}

/**
 * @param  Delays for specific amount of milliseconds
 * @param  millis: Time in milliseconds for delay
 * @retval None
 * @note   Declared as static inline
 */
static __INLINE void Delayms(uint32_t millis) {
     volatile uint32_t timer = TM_Time;

     /* Called from thread */
```

```
	if (!__get_IPSR()) {
		/* Wait for timer to count milliseconds */
		while ((TM_Time - timer) < millis) {
#ifdef DELAY_SLEEP
			/* Go sleep, wait systick interrupt */
			__WFI();
#endif
		}
	} else {
		/* Called from interrupt */
		while (millis) {
			if (SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk) {
				millis--;
			}
		}
	}
}

/**
 * @brief  Initializes timer settings for delay
 * @note   This function will initialize Systick or user timer,
according to settings
 * @param  None
 * @retval None
 */
void TM_DELAY_Init(void);

/**
 * @brief  Gets the TM_Time variable value
 * @param  None
 * @retval Current time in milliseconds
 */
#define TM_DELAY_Time()                          (TM_Time)

/**
 * @brief  Sets value for TM_Time variable
 * @param  time: Time in milliseconds
 * @retval None
 */
#define TM_DELAY_SetTime(time)              (TM_Time = (time))

/**
 * @brief  Re-enables delay timer It has to be configured before
with TM_DELAY_Init()
 * @note   This function enables delay timer. It can be systick
or user selectable timer.
 * @param  None
 * @retval None
 */
void TM_DELAY_EnableDelayTimer(void);

/**
```

```
 * @brief  Disables delay timer
 * @note   This function disables delay timer. It can be systick
or user selectable timer.
 * @param  None
 * @retval None
 */
void TM_DELAY_DisableDelayTimer(void);

/**
 * @brief  Gets the TM_Time2 variable value
 * @param  None
 * @retval Current time in milliseconds
 * @note   This is not meant for public use
 */
#define TM_DELAY_Time2()                    (TM_Time2)

/**
 * @brief  Sets value for TM_Time variable
 * @param  time: Time in milliseconds
 * @retval None
 * @note   This is not meant for public use
 */
#define TM_DELAY_SetTime2(time)             (TM_Time2 = (time))

/**
 * @brief  Creates a new custom timer which has 1ms resolution
 * @note   It uses @ref malloc for memory allocation for timer
structure
 * @param  ReloadValue: Number of milliseconds when timer reaches
zero and callback function is called
 * @param  AutoReload: If set to 1, timer will start again when
it reaches zero and callback is called
 * @param  StartTimer: If set to 1, timer will start immediately
 * @param  *TM_DELAY_CustomTimerCallback: Pointer to callback
function which will be called when timer reaches zero
 * @param  *UserParameters: Pointer to void pointer to user
parameters used as first parameter in callback function
 * @retval Pointer to allocated timer structure
 */
TM_DELAY_Timer_t* TM_DELAY_TimerCreate(uint32_t ReloadValue,
uint8_t AutoReload, uint8_t StartTimer, void
(*TM_DELAY_CustomTimerCallback)(void *), void* UserParameters);

/**
 * @brief  Deletes already allocated timer
 * @param  *Timer: Pointer to @ref TM_DELAY_Timer_t structure
 * @retval None
 */
void TM_DELAY_TimerDelete(TM_DELAY_Timer_t* Timer);

/**
 * @brief  Stops custom timer from counting
```

```
 * @param  *Timer: Pointer to @ref TM_DELAY_Timer_t structure
 * @retval Pointer to @ref TM_DELAY_Timer_t structure
 */
TM_DELAY_Timer_t* TM_DELAY_TimerStop(TM_DELAY_Timer_t* Timer);

/**
 * @brief  Starts custom timer counting
 * @param  *Timer: Pointer to @ref TM_DELAY_Timer_t structure
 * @retval Pointer to @ref TM_DELAY_Timer_t structure
 */
TM_DELAY_Timer_t* TM_DELAY_TimerStart(TM_DELAY_Timer_t* Timer);

/**
 * @brief  Resets custom timer counter value
 * @param  *Timer: Pointer to @ref TM_DELAY_Timer_t structure
 * @retval Pointer to @ref TM_DELAY_Timer_t structure
 */
TM_DELAY_Timer_t* TM_DELAY_TimerReset(TM_DELAY_Timer_t* Timer);

/**
 * @brief  Sets auto reload feature for timer
 * @note   Auto reload features is used for timer which starts
again when zero is reached if auto reload active
 * @param  *Timer: Pointer to @ref TM_DELAY_Timer_t structure
 * uint8_t AutoReload: Set to 1 if you want to enable AutoReload
or 0 to disable
 * @retval Pointer to @ref TM_DELAY_Timer_t structure
 */
TM_DELAY_Timer_t* TM_DELAY_TimerAutoReload(TM_DELAY_Timer_t*
Timer, uint8_t AutoReload);

/**
 * @brief  Sets auto reload value for timer
 * @param  *Timer: Pointer to @ref TM_DELAY_Timer_t structure
 * @param  AutoReloadValue: Value for timer to be set when zero
is reached and callback is called
 * @note   AutoReload feature must be enabled for timer in order
to get this to work properly
 * @retval Pointer to @ref TM_DELAY_Timer_t structure
 */
TM_DELAY_Timer_t* TM_DELAY_TimerAutoReloadValue(TM_DELAY_Timer_t*
Timer, uint32_t AutoReloadValue);

/**
 * @brief  User function, called each 1ms when interrupt from
timer happen
 * @note   Here user should put things which has to be called
periodically
 * @param  None
 * @retval None
 * @note   With __weak parameter to prevent link errors if not
defined by user
```

```
 */
__weak void TM_DELAY_1msHandler(void);


/**
 * @}
 */

/**
 * @}
 */

/**
 * @}
 */

/* C++ detection */
#ifdef __cplusplus
}
#endif

#endif
```

A.2.17. tm_stm32f4_disco

```
/**
 * @author  Tilen Majerle
 * @email   tilen@majerle.eu
 * @website http://stm32f4-discovery.com
 * @link    http://stm32f4-discovery.com/2014/04/stm32f429-
discovery-gpio-tutorial-with-onboard-leds-and-button/
 * @version v1.11
 * @ide     Keil uVision
 * @license GNU GPL v3
 * @brief   Leds and button library for STM32F401 - , STM32F4 -,
STM32F411 - and STM32F429 Discovery boards.
 *          Also works with Nucleo F411 and Nucleo F401 boards
and STM324x9-EVAL boards
 *
@verbatim
   --------------------------------------------------------------
--------
    Copyright (C) Tilen Majerle, 2015

    This program is free software: you can redistribute it and/or
modify
    it under the terms of the GNU General Public License as
published by
    the Free Software Foundation, either version 3 of the
License, or
    any later version.
```

```
         This program is distributed in the hope that it will be
useful,
         but WITHOUT ANY WARRANTY; without even the implied warranty
of
         MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
         GNU General Public License for more details.

         You should have received a copy of the GNU General Public
License
         along with this program.  If not, see
<http://www.gnu.org/licenses/>.
        ---------------------------------------------------------------
--------
      @endverbatim
       */
      #ifndef TM_DISCO_H
      #define TM_DISCO_H 1110
      /**
       * @addtogroup TM_STM32F4xx_Libraries
       * @{
       */

      /**
       * @defgroup TM_DISCO
       * @brief    Leds and buttons for STM32F4 Discovery, Nucleo and
eval boards - http://stm32f4-discovery.com/2014/04/stm32f429-
discovery-gpio-tutorial-with-onboard-leds-and-button/
       * @{
       *
       * Library supports all STM32F4 Discovery boards, All STM32F4
Nucleo boards and STM324x9 eval board.
       *
       * \par Supported boards
       *
       * - STM32F429 Discovery: (STM32F429ZI) -
<code>TM_DISCO_STM32F429_DISCOVERY</code>
       *   - Leds:
       *     - LED_GREEN   on PG13
       *     - LED_RED     on PG14
       *   - Button: (HIGH when pressed)
       *     - Blue button on PA0
       * - NUCLEO-F401: (STM32F401RE) -
<code>TM_DISCO_NUCLEO_F401</code>
       * - NUCLEO-F411: (STM32F411RE) -
<code>TM_DISCO_NUCLEO_F411</code>
       *   - Led:
       *     - LED_GREEN   on PA5
       *   - Button: (LOW when pressed)
       *     - Blue button on PC13
       * - STM32F401-Discovery: (STM32F401VC) -
<code>TM_DISCO_STM32F401_DISCOVERY</code>
```

```
       * - STM32F411-Discovery: (STM32F411VE) -
<code>TM_DISCO_STM32F411_DISCOVERY</code>
       * - STM32F4-Discovery: (STM32F407VG) -
<code>TM_DISCO_STM32F4_DISCOVERY</code>
       *   - Leds:
       *     - LED_GREEN   on PD12
       *     - LED_ORANGE  on PD13
       *     - LED_RED     on PD14
       *     - LED_BLUE    on PD15
       *   - Button: (HIGH when pressed)
       *     - Blue button on PA0
       * - STM324x9-Eval (STM32F439NI) -
<code>TM_DISCO_STM324x9_EVAL</code>
       *   - Leds:
       *     - LED_GREEN   on PG6
       *     - LED_ORANGE  on PG7
       *     - LED_RED     on PG10
       *     - LED_BLUE    on PG12
       *   - Button: (HIGH when pressed)
       *     - Blue button on PA0
       *
       * \par Select your board
       *
       * To select your board, you have several options:
       *   - Add define for your board in defines.h file or
       *   - Add define for your board in compiler's global settings
       *     - For Keil uVision you have "Options for Target" and
"C/C++" tab where you can set this.
       *
       * Imagine, we want to work with STM324x9-Eval board. Then, you
can open <code>defines.h</code> file and add define:
       *
      @verbatim
      //Select STM324x9-Eval for DISCO library
      #define TM_DISCO_STM324x9_EVAL
      @endverbatim
       * Or if you want STM32F429-Discovery, do this:
      @verbatim
      //Select STM32F429-Discovery for DISCO library
      #define TM_DISCO_STM32F429_DISCOVERY
      @endverbatim
       *
       * \par Changelog
       *
      @verbatim
       Version 1.11
        - March 18, 2015
        - Added support for STM324x9-Eval board

       Version 1.10
        - March 14, 2015
        - Fixed issue with pull resistors on boards
```

```
 Version 1.9
  - March 10, 2015
  - Added support for my new GPIO library
  - Added support for STM32F411-Discovery board

 Version 1.8
  - February 01, 2015
  - Added support for button OnPress and OnRelease events

 Version 1.7
  - December 02, 2014
  - Fixed bug with checking if led is on

 Version 1.6
  - November 28, 2014
  - Almost all functions are now defines, for faster execution


 Version 1.5
  - November 06, 2014
  - Added function TM_DISCO_SetLed()

 Version 1.4
  - Added support for Nucleo F411-RE board

 Version 1.3
  - Added support for STM32F401 Discovery board

 Version 1.2
  - Added support for Nucleo F401-RE board

 Version 1.1
  - Check if LED is on or off

 Version 1.0
  - First release
@endverbatim
 *
 * \par Dependencies
 *
@verbatim
 - STM32F4xx
 - defines.h
 - TM GPIO
@endverbatim
 */
#include "stm32f4xx.h"
#include "defines.h"
#include "tm_stm32f4_gpio.h"

/* Recognize correct board */
```

```
	/* CooCox support */
	#if defined(STM32F429_439xx) || defined(STM32F429ZI)
		/* STM32F429 Discovery support */
		#ifndef TM_DISCO_STM32F429_DISCOVERY
			#define TM_DISCO_STM32F429_DISCOVERY
		#endif
	#elif defined(STM32F407VG) || defined(STM32F401VC)// ||
defined(STM32F40_41xxx)
		/* STM32F4 and STM32F401 Discovery support */
		#ifndef TM_DISCO_STM32F4_DISCOVERY
			#define TM_DISCO_STM32F4_DISCOVERY
		#endif
	#elif defined (STM32F401xx) || defined(STM32F401RE) ||
defined(STM32F401RB)
		/* Nucleo F401RE board support */
		#ifndef TM_DISCO_NUCLEO_F401
			#define TM_DISCO_NUCLEO_F401
		#endif
	#elif defined (STM32F411xx) || defined(STM32F411RE) ||
defined(STM32F411RB)
		/* Nucleo F411RE board support */
		#ifndef TM_DISCO_NUCLEO_F411
			#define TM_DISCO_NUCLEO_F411
		#endif
	#endif

	/* STM32F429 Discovery */
	#if defined(TM_DISCO_STM324x9_EVAL)
		#define LED_GREEN					GPIO_PIN_6
		#define LED_ORANGE					GPIO_PIN_7
		#define LED_RED						GPIO_PIN_10
		#define LED_BLUE				GPIO_PIN_12
		#define LED_ALL						LED_GREEN |
LED_RED | LED_ORANGE | LED_BLUE

		#define TM_DISCO_SWAP_LOGIC

		#define TM_DISCO_LED_PORT			GPIOG
		#define TM_DISCO_LED_PINS			LED_GREEN | LED_RED |
LED_ORANGE | LED_BLUE

		#define TM_DISCO_BUTTON_PORT		GPIOA
		#define TM_DISCO_BUTTON_PIN			GPIO_PIN_0
		#define TM_DISCO_BUTTON_PRESSED		1
		#define TM_DISCO_BUTTON_PULL		TM_GPIO_PuPd_DOWN
	#elif defined(TM_DISCO_STM32F429_DISCOVERY)
		#define LED_GREEN					GPIO_PIN_13
		#define LED_RED						GPIO_PIN_14
		#define LED_ORANGE					0
		#define LED_BLUE				0
		#define LED_ALL						LED_GREEN |
LED_RED
```

```
        #define TM_DISCO_LED_PORT             GPIOG
        #define TM_DISCO_LED_PINS             LED_GREEN | LED_RED

        #define TM_DISCO_BUTTON_PORT          GPIOA
        #define TM_DISCO_BUTTON_PIN           GPIO_PIN_0
        #define TM_DISCO_BUTTON_PRESSED       1
        #define TM_DISCO_BUTTON_PULL          TM_GPIO_PuPd_DOWN
    /* STM32F4 & STM32F401 Discovery */
    #elif defined(TM_DISCO_STM32F4_DISCOVERY) ||
defined(TM_DISCO_STM32F401_DISCOVERY) ||
defined(TM_DISCO_STM32F411_DISCOVERY)
        #define LED_GREEN                          GPIO_PIN_12
        #define LED_ORANGE                         GPIO_PIN_13
        #define LED_RED                            GPIO_PIN_14
        #define LED_BLUE                      GPIO_PIN_15
        #define LED_ALL                            LED_GREEN |
LED_RED | LED_ORANGE | LED_BLUE

        #define TM_DISCO_LED_PORT             GPIOD
        #define TM_DISCO_LED_PINS             LED_GREEN | LED_RED |
LED_ORANGE | LED_BLUE

        #define TM_DISCO_BUTTON_PORT          GPIOA
        #define TM_DISCO_BUTTON_PIN           GPIO_PIN_0
        #define TM_DISCO_BUTTON_PRESSED       1
        #define TM_DISCO_BUTTON_PULL          TM_GPIO_PuPd_DOWN
    /* Nucleo F401-RE & F411-RE */
    #elif defined(TM_DISCO_NUCLEO_F401) ||
defined(TM_DISCO_NUCLEO_F411)
        #define LED_GREEN                          GPIO_PIN_5
        #define LED_RED                            0
        #define LED_ORANGE                         0
        #define LED_BLUE                      0
        #define LED_ALL                            LED_GREEN

        #define TM_DISCO_LED_PORT             GPIOA
        #define TM_DISCO_LED_PINS             LED_GREEN

        #define TM_DISCO_BUTTON_PORT          GPIOC
        #define TM_DISCO_BUTTON_PIN           GPIO_PIN_13
        #define TM_DISCO_BUTTON_PRESSED       0
        #define TM_DISCO_BUTTON_PULL          TM_GPIO_PuPd_UP
    /* STM324x9 EVAL board */
    #else
        #error "tm_stm32f4_disco.h: Please select your board. Open
tm_stm32f4_disco.h and follow instructions!!"
    #endif

    /**
     * @defgroup TM_DISCO_Functions
     * @brief    Library Functions
```

```
 * @{
 */

/**
 * @brief  Configures LED pins as outputs
 * @param  None
 * @retval None
 */
void TM_DISCO_LedInit(void);

/**
 * @brief  Configures Button pin as input
 * @param  None
 * @retval None
 */
void TM_DISCO_ButtonInit(void);

/**
 * @brief  Turns on LED on board
 * @note   STM32F4x9-Eval board uses inverse logic for leds
 * @param  led: LED you want to turn on
 *            - LED_RED: Red led
 *            - LED_GREEN: Green led
 *            - LED_BLUE: Blue led
 *            - LED_ORANGE: Orange led
 *            - LED_ALL: All leds
 * @retval None
 */
#ifndef TM_DISCO_SWAP_LOGIC
     #define TM_DISCO_LedOn(led)
TM_GPIO_SetPinHigh(TM_DISCO_LED_PORT, (led))
#else
     #define TM_DISCO_LedOn(led)
TM_GPIO_SetPinLow(TM_DISCO_LED_PORT, (led))
#endif

/**
 * @brief  Turns off LED on board
 * @note   STM32F4x9-Eval board uses inverse logic for leds
 * @param  led: LED you want to turn off
 *            - LED_RED: Red led
 *            - LED_GREEN: Green led
 *            - LED_BLUE: Blue led
 *            - LED_ORANGE: Orange led
 *            - LED_ALL: All leds
 * @retval None
 */
#ifndef TM_DISCO_SWAP_LOGIC
     #define TM_DISCO_LedOff(led)
TM_GPIO_SetPinLow(TM_DISCO_LED_PORT, (led))
#else
```

```
        #define TM_DISCO_LedOff(led)
TM_GPIO_SetPinHigh(TM_DISCO_LED_PORT, (led))
    #endif

    /**
     * @brief  Toggles LED on board
     * @param  led: LED you want to toggle
     *            - LED_RED: Red led
     *            - LED_GREEN: Green led
     *            - LED_BLUE: Blue led
     *            - LED_ORANGE: Orange led
     *            - LED_ALL: All leds
     * @retval None
     */
    #define TM_DISCO_LedToggle(led)
TM_GPIO_TogglePinValue(TM_DISCO_LED_PORT, (led))

    /**
     * @brief  Checks if led is on
     * @note   STM32F4x9-Eval board uses inverse logic for leds
     * @param  led: Led you want to checking
     *            - LED_RED: Red led
     *            - LED_GREEN: Green led
     *            - LED_BLUE: Blue led
     *            - LED_ORANGE: Orange led
     *            - LED_ALL: All leds
     * @retval 1 if led is on or 0 if not
     */
    #ifndef TM_DISCO_SWAP_LOGIC
    #define TM_DISCO_LedIsOn(led)
TM_GPIO_GetOutputPinValue(TM_DISCO_LED_PORT, (led))
    #else
    #define TM_DISCO_LedIsOn(led)
!TM_GPIO_GetOutputPinValue(TM_DISCO_LED_PORT, (led))
    #endif

    /**
     * @brief  Sets led value
     * @param  led: LED you want to set value
     *            - LED_RED: Red led
     *            - LED_GREEN: Green led
     *            - LED_BLUE: Blue led
     *            - LED_ORANGE: Orange led
     *            - LED_ALL: All leds
     * @param  state: Set or clear led
     *            - 0: led is off
     *            - > 0: led is on
     * @retval None
     */
    #define TM_DISCO_SetLed(led, state)    ((state) ?
TM_DISCO_LedOn(led): TM_DISCO_LedOff(led))
```

```
/**
 * @brief  Checks if user button is pressed
 * @param  None
 * @retval Button status
 *            - 0: Button is not pressed
 *            - > 0: Button is pressed
 */
#define TM_DISCO_ButtonPressed()
((TM_GPIO_GetInputPinValue(TM_DISCO_BUTTON_PORT, TM_DISCO_BUTTON_PIN)
== 0) != TM_DISCO_BUTTON_PRESSED)

/**
 * @brief  Checks if button was pressed now, but was not already
pressed before
 * @param  None
 * @retval Button on pressed value
 *           - 0: In case that button has been already pressed on
last call or was not pressed at all yet
 *           - > 0: Button was pressed, but state before was
released
 */
uint8_t TM_DISCO_ButtonOnPressed(void);

/**
 * @brief  Checks if button was released now, but was already
pressed before
 * @param  None
 * @retval Button on released value
 *           - 0: In case that button has been already released
on last call or was not released at all yet
 *           - > 0: Button was released, but state before was
pressed
 */
uint8_t TM_DISCO_ButtonOnReleased(void);

/**
 * @}
 */

/**
 * @}
 */

/**
 * @}
 */

#endif
```

A.2.18. tm_stm32f4_gpio

```
/**
 * @author  Tilen Majerle
 * @email   tilen@majerle.eu
 * @website http://stm32f4-discovery.com
 * @link    http://stm32f4-discovery.com/2015/03/library-53-gpio-
for-stm32f4
 * @version v1.5
 * @ide     Keil uVision
 * @license GNU GPL v3
 * @brief   GPIO Library for STM32F4xx devices
 *
@verbatim
   ----------------------------------------------------------------------
--------
    Copyright (C) Tilen Majerle, 2015

    This program is free software: you can redistribute it and/or
modify
    it under the terms of the GNU General Public License as
published by
    the Free Software Foundation, either version 3 of the
License, or
    any later version.

    This program is distributed in the hope that it will be
useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty
of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public
License
    along with this program.  If not, see
<http://www.gnu.org/licenses/>.
   ----------------------------------------------------------------------
--------
@endverbatim
 */
#ifndef TM_GPIO_H
#define TM_GPIO_H 150

/* C++ detection */
#ifdef __cplusplus
extern "C" {
#endif

/**
 * @addtogroup TM_STM32F4xx_Libraries
```

```
 * @{
 */

/**
 * @defgroup TM_GPIO
 * @brief    TM GPIO Library for STM32F4xx - http://stm32f4-
discovery.com/2015/03/library-53-gpio-for-stm32f4
 * @{
 *
 * GPIO library can be used for GPIO pins.
 *
 * It features fast initialization methods as well pin
input/output methods.
 *
 * It can be used as replacement for STD/HAL drivers GPIO
library.
 *
 * \par Changelog
 *
@verbatim
 Version 1.5
  - June 10 2015
  - Added 2 new functions for getting used GPIO pins

 Version 1.4
  - April 28, 2015
  - Added support for PORT locking

 Version 1.3
  - March 23, 2015
  - Totally independent from HAL / SPD drivers
  - Library can be used with any drivers or totally itself

 Version 1.2
  - March 10, 2015
  - Added functions TM_GPIO_SetPinAsInput and
TM_GPIO_SetPinAsOutput
  - Added functions TM_GPIO_GetPortSource and
TM_GPIO_GetPinSource
0
 Version 1.1
  - March 09, 2015
  - Added function to deinit pin. Pin is set to analog input
which allows lowest current consumption

 Version 1.0
  - March 08, 2015
  - Initial release
@endverbatim
 *
 * \par Dependencies
 *
```

```
@verbatim
 - STM32F4xx
 - STM32F4xx GPIO
 - defines.h
@endverbatim
 */

#include "stm32f4xx.h"
#include "stm32f4xx_gpio.h"
#include "defines.h"

/**
 * @defgroup TM_GPIO_Macros
 * @brief    GPIO Library macros
 * @{
 */

/**
 * @brief GPIO Pins declarations
 * @note  For HAL drivers compatibility
 */

#ifndef GPIO_PIN_0
#define GPIO_PIN_0          ((uint16_t)0x0001)
#define GPIO_PIN_1          ((uint16_t)0x0002)
#define GPIO_PIN_2          ((uint16_t)0x0004)
#define GPIO_PIN_3          ((uint16_t)0x0008)
#define GPIO_PIN_4          ((uint16_t)0x0010)
#define GPIO_PIN_5          ((uint16_t)0x0020)
#define GPIO_PIN_6          ((uint16_t)0x0040)
#define GPIO_PIN_7          ((uint16_t)0x0080)
#define GPIO_PIN_8          ((uint16_t)0x0100)
#define GPIO_PIN_9          ((uint16_t)0x0200)
#define GPIO_PIN_10         ((uint16_t)0x0400)
#define GPIO_PIN_11         ((uint16_t)0x0800)
#define GPIO_PIN_12         ((uint16_t)0x1000)
#define GPIO_PIN_13         ((uint16_t)0x2000)
#define GPIO_PIN_14         ((uint16_t)0x4000)
#define GPIO_PIN_15         ((uint16_t)0x8000)
#define GPIO_PIN_ALL  ((uint16_t)0xFFFF)
#endif

/**
 * @brief GPIO Pins declarations
 * @note  For STD Periph drivers compatibility
 */

#ifndef GPIO_Pin_0
#define GPIO_Pin_0          ((uint16_t)0x0001)
#define GPIO_Pin_1          ((uint16_t)0x0002)
#define GPIO_Pin_2          ((uint16_t)0x0004)
#define GPIO_Pin_3          ((uint16_t)0x0008)
```

```
#define GPIO_Pin_4          ((uint16_t)0x0010)
#define GPIO_Pin_5          ((uint16_t)0x0020)
#define GPIO_Pin_6          ((uint16_t)0x0040)
#define GPIO_Pin_7          ((uint16_t)0x0080)
#define GPIO_Pin_8          ((uint16_t)0x0100)
#define GPIO_Pin_9          ((uint16_t)0x0200)
#define GPIO_Pin_10         ((uint16_t)0x0400)
#define GPIO_Pin_11         ((uint16_t)0x0800)
#define GPIO_Pin_12         ((uint16_t)0x1000)
#define GPIO_Pin_13         ((uint16_t)0x2000)
#define GPIO_Pin_14         ((uint16_t)0x4000)
#define GPIO_Pin_15         ((uint16_t)0x8000)
#define GPIO_Pin_All  ((uint16_t)0xFFFF)
#endif

/**
 * @}
 */

/**
 * @defgroup TM_GPIO_Typedefs
 * @brief    GPIO Typedefs used for GPIO library for
initialization purposes
 * @{
 */

/**
 * @brief GPIO Mode enumeration
 */
typedef enum {
     TM_GPIO_Mode_IN = 0x00,  /*!< GPIO Pin as General Purpose
Input */
     TM_GPIO_Mode_OUT = 0x01, /*!< GPIO Pin as General Purpose
Output */
     TM_GPIO_Mode_AF = 0x02,  /*!< GPIO Pin as Alternate
Function */
     TM_GPIO_Mode_AN = 0x03,  /*!< GPIO Pin as Analog */
} TM_GPIO_Mode_t;

/**
 * @brief GPIO Output type enumeration
 */
typedef enum {
     TM_GPIO_OType_PP = 0x00, /*!< GPIO Output Type Push-Pull */
     TM_GPIO_OType_OD = 0x01  /*!< GPIO Output Type Open-Drain
*/
} TM_GPIO_OType_t;

/**
 * @brief  GPIO Speed enumeration
 */
typedef enum {
```

```
        TM_GPIO_Speed_Low = 0x00,    /*!< GPIO Speed Low */
        TM_GPIO_Speed_Medium = 0x01, /*!< GPIO Speed Medium */
        TM_GPIO_Speed_Fast = 0x02,   /*!< GPIO Speed Fast */
        TM_GPIO_Speed_High = 0x03    /*!< GPIO Speed High */
    } TM_GPIO_Speed_t;

    /**
     * @brief GPIO pull resistors enumeration
     */
    typedef enum {
        TM_GPIO_PuPd_NOPULL = 0x00, /*!< No pull resistor */
        TM_GPIO_PuPd_UP = 0x01,     /*!< Pull up resistor enabled
*/
        TM_GPIO_PuPd_DOWN = 0x02    /*!< Pull down resistor enabled
*/
    } TM_GPIO_PuPd_t;

    /**
     * @} TM_GPIO_Typedefs
     */

    /**
     * @defgroup TM_GPIO_Functions
     * @brief    GPIO Functions
     * @{
     */

    /**
     * @brief  Initializes GPIO pins(s)
     * @note   This function also enables clock for GPIO port
     * @param  GPIOx: Pointer to GPIOx port you will use for
initialization
     * @param  GPIO_Pin: GPIO pin(s) you will use for initialization
     * @param  GPIO_Mode: Select GPIO mode. This parameter can be a
value of @ref TM_GPIO_Mode_t enumeration
     * @param  GPIO_OType: Select GPIO Output type. This parameter
can be a value of @ref TM_GPIO_OType_t enumeration
     * @param  GPIO_PuPd: Select GPIO pull resistor. This parameter
can be a value of @ref TM_GPIO_PuPd_t enumeration
     * @param  GPIO_Speed: Select GPIO speed. This parameter can be a
value of @ref TM_GPIO_Speed_t enumeration
     * @retval None
     */
    void TM_GPIO_Init(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin,
TM_GPIO_Mode_t GPIO_Mode, TM_GPIO_OType_t GPIO_OType, TM_GPIO_PuPd_t
GPIO_PuPd, TM_GPIO_Speed_t GPIO_Speed);

    /**
     * @brief  Initializes GPIO pins(s) as alternate function
     * @note   This function also enables clock for GPIO port
     * @param  GPIOx: Pointer to GPIOx port you will use for
initialization
```

```
 * @param  GPIO_Pin: GPIO pin(s) you will use for initialization
 * @param  GPIO_OType: Select GPIO Output type. This parameter can be a value of @ref TM_GPIO_OType_t enumeration
 * @param  GPIO_PuPd: Select GPIO pull resistor. This parameter can be a value of @ref TM_GPIO_PuPd_t enumeration
 * @param  GPIO_Speed: Select GPIO speed. This parameter can be a value of @ref TM_GPIO_Speed_t enumeration
 * @param  Alternate: Alternate function you will use
 * @retval None
 */
void TM_GPIO_InitAlternate(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, TM_GPIO_OType_t GPIO_OType, TM_GPIO_PuPd_t GPIO_PuPd, TM_GPIO_Speed_t GPIO_Speed, uint8_t Alternate);

/**
 * @brief  Deinitializes pin(s)
 * @note   Pins(s) will be set as analog mode to get low power consumption
 * @param  GPIOx: GPIOx PORT where you want to set pin as input
 * @param  GPIO_Pin: Select GPIO pin(s). You can select more pins with | (OR) operator to set them as input
 * @retval None
 */
void TM_GPIO_DeInit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);

/**
 * @brief  Sets pin(s) as input
 * @note   Pins HAVE to be initialized first using @ref TM_GPIO_Init() or @ref TM_GPIO_InitAlternate() function
 * @note   This is just an option for fast input mode
 * @param  GPIOx: GPIOx PORT where you want to set pin as input
 * @param  GPIO_Pin: Select GPIO pin(s). You can select more pins with | (OR) operator to set them as input
 * @retval None
 */
void TM_GPIO_SetPinAsInput(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);

/**
 * @brief  Sets pin(s) as output
 * @note   Pins HAVE to be initialized first using @ref TM_GPIO_Init() or @ref TM_GPIO_InitAlternate() function
 * @note   This is just an option for fast output mode
 * @param  GPIOx: GPIOx PORT where you want to set pin as output
 * @param  GPIO_Pin: Select GPIO pin(s). You can select more pins with | (OR) operator to set them as output
 * @retval None
 */
void TM_GPIO_SetPinAsOutput(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);

/**
```

```
      * @brief  Sets pin(s) as analog
      * @note   Pins HAVE to be initialized first using @ref
TM_GPIO_Init() or @ref TM_GPIO_InitAlternate() function
      * @note   This is just an option for fast analog mode
      * @param  GPIOx: GPIOx PORT where you want to set pin as analog
      * @param  GPIO_Pin: Select GPIO pin(s). You can select more pins
with | (OR) operator to set them as analog
      * @retval None
      */
     void TM_GPIO_SetPinAsAnalog(GPIO_TypeDef* GPIOx, uint16_t
GPIO_Pin);

     /**
      * @brief  Sets pin(s) as alternate function
      * @note   For proper alternate function, you should first init
pin using @ref TM_GPIO_InitAlternate() function.
      *            This functions is only used for changing GPIO mode
      * @param  GPIOx: GPIOx PORT where you want to set pin as
alternate
      * @param  GPIO_Pin: Select GPIO pin(s). You can select more pins
with | (OR) operator to set them as alternate
      * @retval None
      */
     void TM_GPIO_SetPinAsAlternate(GPIO_TypeDef* GPIOx, uint16_t
GPIO_Pin);

     /**
      * @brief  Sets pull resistor settings to GPIO pin(s)
      * @note   Pins HAVE to be initialized first using @ref
TM_GPIO_Init() or @ref TM_GPIO_InitAlternate() function
      * @param  *GPIOx: GPIOx PORT where you want to select pull
resistor
      * @param  GPIO_Pin: Select GPIO pin(s). You can select more pins
with | (OR) operator to set them as output
      * @param  GPIO_PuPd: Pull resistor option. This parameter can be
a value of @ref TM_GPIO_PuPd_t enumeration
      * @retval None
      */
     void TM_GPIO_SetPullResistor(GPIO_TypeDef* GPIOx, uint16_t
GPIO_Pin, TM_GPIO_PuPd_t GPIO_PuPd);

     /**
      * @brief  Sets pin(s) low
      * @note   Defined as macro to get maximum speed using register
access
      * @param  GPIOx: GPIOx PORT where you want to set pin low
      * @param  GPIO_Pin: Select GPIO pin(s). You can select more pins
with | (OR) operator to set them low
      * @retval None
      */
     #define TM_GPIO_SetPinLow(GPIOx, GPIO_Pin)            ((GPIOx)-
>BSRRH = (GPIO_Pin))
```

```
    /**
     * @brief  Sets pin(s) high
     * @note   Defined as macro to get maximum speed using register
access
     * @param  GPIOx: GPIOx PORT where you want to set pin high
     * @param  GPIO_Pin: Select GPIO pin(s). You can select more pins
with | (OR) operator to set them high
     * @retval None
     */
    #define TM_GPIO_SetPinHigh(GPIOx, GPIO_Pin)            ((GPIOx)-
>BSRRL = (GPIO_Pin))

    /**
     * @brief  Sets pin(s) value
     * @note   Defined as macro to get maximum speed using register
access
     * @param  GPIOx: GPIOx PORT where you want to set pin value
     * @param  GPIO_Pin: Select GPIO pin(s). You can select more pins
with | (OR) operator to set them value
     * @param  val: If parameter is 0 then pin will be low, otherwise
high
     * @retval None
     */
    #define TM_GPIO_SetPinValue(GPIOx, GPIO_Pin, val)((val) ?
TM_GPIO_SetPinHigh(GPIOx, GPIO_Pin) : TM_GPIO_SetPinLow(GPIOx,
GPIO_Pin))

    /**
     * @brief  Toggles pin(s)
     * @note   Defined as macro to get maximum speed using register
access
     * @param  GPIOx: GPIOx PORT where you want to toggle pin value
     * @param  GPIO_Pin: Select GPIO pin(s). You can select more pins
with | (OR) operator to toggle them all at a time
     * @retval None
     */
    #define TM_GPIO_TogglePinValue(GPIOx, GPIO_Pin)        ((GPIOx)-
>ODR ^= (GPIO_Pin))

    /**
     * @brief  Sets value to entire GPIO PORT
     * @note   Defined as macro to get maximum speed using register
access
     * @param  GPIOx: GPIOx PORT where you want to set value
     * @param  value: Value for GPIO OUTPUT data
     * @retval None
     */
    #define TM_GPIO_SetPortValue(GPIOx, value)             ((GPIOx)-
>ODR = (value))

    /**
```

```
 * @brief  Gets input data bit
 * @note   Defined as macro to get maximum speed using register
access
 * @param  GPIOx: GPIOx PORT where you want to read input bit
value
 * @param  GPIO_Pin: GPIO pin where you want to read value
 * @retval 1 in case pin is high, or 0 if low
 */
#define TM_GPIO_GetInputPinValue(GPIOx, GPIO_Pin)(((GPIOx)->IDR &
(GPIO_Pin)) == 0 ? 0 : 1)

/**
 * @brief  Gets output data bit
 * @note   Defined as macro to get maximum speed using register
access
 * @param  GPIOx: GPIOx PORT where you want to read output bit
value
 * @param  GPIO_Pin: GPIO pin where you want to read value
 * @retval 1 in case pin is high, or 0 if low
 */
#define TM_GPIO_GetOutputPinValue(GPIOx, GPIO_Pin)     (((GPIOx)-
>ODR & (GPIO_Pin)) == 0 ? 0 : 1)

/**
 * @brief  Gets input value from entire GPIO PORT
 * @note   Defined as macro to get maximum speed using register
access
 * @param  GPIOx: GPIOx PORT where you want to read input data
value
 * @retval Entire PORT INPUT register
 */
#define TM_GPIO_GetPortInputValue(GPIOx)               ((GPIOx)-
>IDR)

/**
 * @brief  Gets output value from entire GPIO PORT
 * @note   Defined as macro to get maximum speed using register
access
 * @param  GPIOx: GPIOx PORT where you want to read output data
value
 * @retval Entire PORT OUTPUT register
 */
#define TM_GPIO_GetPortOutputValue(GPIOx)              ((GPIOx)-
>ODR)

/**
 * @brief  Gets port source from desired GPIOx PORT
 * @note   Meant for private use, unless you know what are you
doing
 * @param  GPIOx: GPIO PORT for calculating port source
 * @retval Calculated port source for GPIO
 */
```

```
uint16_t TM_GPIO_GetPortSource(GPIO_TypeDef* GPIOx);

/**
 * @brief  Gets pin source from desired GPIO pin
 * @note   Meant for private use, unless you know what are you
doing
 * @param  GPIO_Pin: GPIO pin for calculating port source
 * @retval Calculated pin source for GPIO pin
 */
uint16_t TM_GPIO_GetPinSource(uint16_t GPIO_Pin);

/**
 * @brief  Locks GPIOx register for future changes
 * @note   You are not able to config GPIO registers until new
MCU reset occurs
 * @param  *GPIOx: GPIOx PORT where you want to lock config
registers
 * @param  GPIO_Pin: GPIO pin(s) where you want to lock config
registers
 * @retval None
 */
void TM_GPIO_Lock(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);

/**
 * @brief  Gets bit separated pins which were used at least once
in library and were not deinitialized
 * @param  *GPIOx: Pointer to GPIOx peripheral where to check
used GPIO pins
 * @retval Bit values for used pins
 */
uint16_t TM_GPIO_GetUsedPins(GPIO_TypeDef* GPIOx);

/**
 * @brief  Gets bit separated pins which were not used at in
library or were deinitialized
 * @param  *GPIOx: Pointer to GPIOx peripheral where to check
used GPIO pins
 * @retval Bit values for free pins
 */
uint16_t TM_GPIO_GetFreePins(GPIO_TypeDef* GPIOx);

/**
 * @}
 */
/**
 * @}
 */
/**
 * @}
 */

/* C++ detection */
```

```
#ifdef __cplusplus
}
#endif

#endif
```

A.2.19. tm_stm32f4_spi

```
/**
 * @author  Tilen Majerle
 * @email   tilen@majerle.eu
 * @website http://stm32f4-discovery.com
 * @link    http://stm32f4-discovery.com/2014/04/library-05-spi-
for-stm32f4xx/
 * @version v2.0
 * @ide     Keil uVision
 * @license GNU GPL v3
 * @brief   SPI library for STM32F4xx
 *
@verbatim
   ----------------------------------------------------------------
--------
    Copyright (C) Tilen Majerle, 2015

    This program is free software: you can redistribute it and/or
modify
    it under the terms of the GNU General Public License as
published by
    the Free Software Foundation, either version 3 of the
License, or
    any later version.

    This program is distributed in the hope that it will be
useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty
of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public
License
    along with this program.  If not, see
<http://www.gnu.org/licenses/>.
@endverbatim
 */
#ifndef TM_SPI_H
#define TM_SPI_H 200

/* C++ detection */
#ifdef __cplusplus
extern "C" {
```

```
#endif

/**
 * @addtogroup TM_STM32F4xx_Libraries
 * @{
 */

/**
 * @defgroup TM_SPI
 * @brief    SPI library for STM32F4xx - http://stm32f4-
discovery.com/2014/04/library-05-spi-for-stm32f4xx/
 * @{
 *
 * It supports all 6 SPIs in master with 3 Lines Full Duplex mode
 *
 * \par Default SPI settings
 *
 * All six SPIs work the same principle by default:
 *    - Master mode
 *    - 8 data bits
 *    - Clock polarity low, data sampled at first edge, SPI mode
0
 *    - Prescaler set to 32
 *    - Firstbit MSB
 *    - Software SS pin configure
 *    - Direction is full duplex 3 wires
 *
 *    \par Pinout
 *
@verbatim
        |PINS PACK 1            |PINS PACK 2            |PINS
PACK 3
SPIX    |MOSI   MISO    SCK     |MOSI   MISO    SCK     |MOSI
MISO    SCK
        |
SPI1    |PA7    PA6     PA5     |PB5    PB4     PB3     |
SPI2    |PC3    PC2     PB10    |PB15   PB14    PB13    |PI3
PI2     PI0
SPI3    |PB5    PB4     PB3     |PC12   PC11    PC10    |
SPI4    |PE6    PE5     PE2     |PE14   PE13    PE12    |
SPI5    |PF9    PF8     PF7     |PF11   PH7     PH6     |
SPI6    |PG14   PG12    PG13    |
@endverbatim
 *
 *    In case these pins are not good for you, you can use
 *    TM_SPI_PinsPack_Custom in function and callback function
will be called,
 *    where you can initialize your custom pinout for your SPI
peripheral
 *
 *    Possible changes to each SPI. Set this defines in your
defines.h file.
```

```
 *
 *    Change x with 1-6, to match your SPI
 *
@verbatim
//Default prescaler
#define TM_SPIx_PRESCALER   SPI_BaudRatePrescaler_32
//Specify datasize
#define TM_SPIx_DATASIZE    SPI_DataSize_8b
//Specify which bit is first
#define TM_SPIx_FIRSTBIT    SPI_FirstBit_MSB
//Mode, master or slave
#define TM_SPIx_MASTERSLAVE SPI_Mode_Master
//Specify mode of operation, clock polarity and clock phase
#define TM_SPIx_MODE        TM_SPI_Mode_0
@endverbatim
 *
 * \par Changelog
 *
@verbatim
 - Version 2.0
  - June 06, 2015
  - Added support for changing SPI data size on runtime

 Version 1.9
  - March 21, 2015
  - SPI Send BUG fixed

 Version 1.8
  - March 10, 2015
  - Updated to be mode independent of STD/HAL drivers

 Version 1.7
  - March 08, 2015
  - Added support for my new GPIO settings

 Version 1.6
  - March 05, 2015
  - Added 2 new functions, TM_SPI_InitFull and
TM_SPI_GetPrescalerFromMaxFrequency()

 Version 1.5
  - January 13, 2015
  - Added function TM_SPI_InitWithMode() to initialize SPI with
custom SPI mode on the fly

 Version 1.4
  - November 09, 2014
  - Added methods for 16-bit SPI mode

 Version 1.3
  - September 14, 2014
  - Added additional pins for SPI2
```

```
 Version 1.0
  - First release
@endverbatim
 *
 * \par Dependencies
 *
@verbatim
 - STM32F4xx
 - STM32F4xx RCC
 - STM32F4xx GPIO
 - STM32F4xx SPI
 - defines.h
 - attributes.h
 - TM GPIO
@endverbatim
 */

#include "stm32f4xx.h"
#include "stm32f4xx_rcc.h"
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_spi.h"
#include "defines.h"
#include "attributes.h"
#include "tm_stm32f4_gpio.h"

/**
 * @defgroup TM_SPI_Typedefs
 * @brief    Library Typedefs
 * @{
 */

/**
 * @brief  SPI modes selection
 */
typedef enum {
     TM_SPI_Mode_0, /*!< Clock polarity low, clock phase 1st
edge */
     TM_SPI_Mode_1, /*!< Clock polarity low, clock phase 2nd
edge */
     TM_SPI_Mode_2, /*!< Clock polarity high, clock phase 1st
edge */
     TM_SPI_Mode_3  /*!< Clock polarity high, clock phase 2nd
edge */
} TM_SPI_Mode_t;

/**
 * @brief  SPI PinsPack enumeration to select pins combination
for SPI
 */
typedef enum {
```

```
           TM_SPI_PinsPack_1,       /*!< Select PinsPack1 from Pinout
table for specific SPI */
           TM_SPI_PinsPack_2,       /*!< Select PinsPack2 from Pinout
table for specific SPI */
           TM_SPI_PinsPack_3,       /*!< Select PinsPack3 from Pinout
table for specific SPI */
           TM_SPI_PinsPack_Custom   /*!< Select custom pins for
specific SPI, callback will be called, look @ref
TM_SPI_InitCustomPinsCallback */
      } TM_SPI_PinsPack_t;

      /**
       * @brief  Daza size enumeration
       */
      typedef enum {
           TM_SPI_DataSize_8b, /*!< SPI in 8-bits mode */
           TM_SPI_DataSize_16b /*!< SPI in 16-bits mode */
      } TM_SPI_DataSize_t;

      /**
       * @}
       */

       /**
       * @defgroup TM_SPI_Macros
       * @brief    Library defines
       * @{
       */

      /**
       * @brief  Supported SPI modules
       */
      #define USE_SPI1
      #define USE_SPI2
      #define USE_SPI3
      #ifdef SPI4
      #define USE_SPI4
      #else
      #warning "SPI4 undefined. Please update library with STD drivers
from ST.com"
      #endif
      #ifdef SPI5
      #define USE_SPI5
      #else
      #warning "SPI5 undefined. Please update library with STD drivers
from ST.com"
      #endif
      #ifdef SPI6
      #define USE_SPI6
      #else
      #warning "SPI6 undefined. Please update library with STD drivers
from ST.com"
```

```
#endif

//----- SPI1 options start -------
//Options can be overwriten in defines.h file
#ifndef TM_SPI1_PRESCALER
#define TM_SPI1_PRESCALER  SPI_BaudRatePrescaler_32
#endif
//Specify datasize
#ifndef TM_SPI1_DATASIZE
#define TM_SPI1_DATASIZE   SPI_DataSize_8b
#endif
//Specify which bit is first
#ifndef TM_SPI1_FIRSTBIT
#define TM_SPI1_FIRSTBIT   SPI_FirstBit_MSB
#endif
//Mode, master or slave
#ifndef TM_SPI1_MASTERSLAVE
#define TM_SPI1_MASTERSLAVE SPI_Mode_Master
#endif
//Specify mode of operation, clock polarity and clock phase
#ifndef TM_SPI1_MODE
#define TM_SPI1_MODE       TM_SPI_Mode_0
#endif
//----- SPI1 options end -------

//----- SPI2 options start -------
//Options can be overwriten in defines.h file
#ifndef TM_SPI2_PRESCALER
#define TM_SPI2_PRESCALER  SPI_BaudRatePrescaler_32
#endif
//Specify datasize
#ifndef TM_SPI2_DATASIZE
#define TM_SPI2_DATASIZE   SPI_DataSize_8b
#endif
//Specify which bit is first
#ifndef TM_SPI2_FIRSTBIT
#define TM_SPI2_FIRSTBIT   SPI_FirstBit_MSB
#endif
//Mode, master or slave
#ifndef TM_SPI2_MASTERSLAVE
#define TM_SPI2_MASTERSLAVE SPI_Mode_Master
#endif
//Specify mode of operation, clock polarity and clock phase
#ifndef TM_SPI2_MODE
#define TM_SPI2_MODE       TM_SPI_Mode_0
#endif
//----- SPI2 options end -------

//----- SPI3 options start -------
//Options can be overwriten in defines.h file
#ifndef TM_SPI3_PRESCALER
#define TM_SPI3_PRESCALER  SPI_BaudRatePrescaler_32
```

```
#endif
//Specify datasize
#ifndef TM_SPI3_DATASIZE
#define TM_SPI3_DATASIZE   SPI_DataSize_8b
#endif
//Specify which bit is first
#ifndef TM_SPI3_FIRSTBIT
#define TM_SPI3_FIRSTBIT   SPI_FirstBit_MSB
#endif
//Mode, master or slave
#ifndef TM_SPI3_MASTERSLAVE
#define TM_SPI3_MASTERSLAVE SPI_Mode_Master
#endif
//Specify mode of operation, clock polarity and clock phase
#ifndef TM_SPI3_MODE
#define TM_SPI3_MODE       TM_SPI_Mode_0
#endif
//----- SPI3 options end -------

//----- SPI4 options start -------
//Options can be overwriten in defines.h file
#ifndef TM_SPI4_PRESCALER
#define TM_SPI4_PRESCALER  SPI_BaudRatePrescaler_32
#endif
//Specify datasize
#ifndef TM_SPI4_DATASIZE
#define TM_SPI4_DATASIZE   SPI_DataSize_8b
#endif
//Specify which bit is first
#ifndef TM_SPI4_FIRSTBIT
#define TM_SPI4_FIRSTBIT   SPI_FirstBit_MSB
#endif
//Mode, master or slave
#ifndef TM_SPI4_MASTERSLAVE
#define TM_SPI4_MASTERSLAVE SPI_Mode_Master
#endif
//Specify mode of operation, clock polarity and clock phase
#ifndef TM_SPI4_MODE
#define TM_SPI4_MODE       TM_SPI_Mode_0
#endif
//----- SPI4 options end -------

//----- SPI5 options start -------
//Options can be overwriten in defines.h file
#ifndef TM_SPI5_PRESCALER
#define TM_SPI5_PRESCALER  SPI_BaudRatePrescaler_32
#endif
//Specify datasize
#ifndef TM_SPI5_DATASIZE
#define TM_SPI5_DATASIZE   SPI_DataSize_8b
#endif
//Specify which bit is first
```

```
#ifndef TM_SPI5_FIRSTBIT
#define TM_SPI5_FIRSTBIT   SPI_FirstBit_MSB
#endif
//Mode, master or slave
#ifndef TM_SPI5_MASTERSLAVE
#define TM_SPI5_MASTERSLAVE SPI_Mode_Master
#endif
//Specify mode of operation, clock polarity and clock phase
#ifndef TM_SPI5_MODE
#define TM_SPI5_MODE       TM_SPI_Mode_0
#endif
//----- SPI5 options end -------

//----- SPI6 options start -------
//Options can be overwriten in defines.h file
#ifndef TM_SPI6_PRESCALER
#define TM_SPI6_PRESCALER  SPI_BaudRatePrescaler_32
#endif
//Specify datasize
#ifndef TM_SPI6_DATASIZE
#define TM_SPI6_DATASIZE   SPI_DataSize_8b
#endif
//Specify which bit is first
#ifndef TM_SPI6_FIRSTBIT
#define TM_SPI6_FIRSTBIT   SPI_FirstBit_MSB
#endif
//Mode, master or slave
#ifndef TM_SPI6_MASTERSLAVE
#define TM_SPI6_MASTERSLAVE SPI_Mode_Master
#endif
//Specify mode of operation, clock polarity and clock phase
#ifndef TM_SPI6_MODE
#define TM_SPI6_MODE       TM_SPI_Mode_0
#endif
//----- SPI6 options end -------

/**
 * @brief  Check SPI busy status
 */
#define SPI_IS_BUSY(SPIx) (((SPIx)->SR & (SPI_SR_TXE | 
SPI_SR_RXNE)) == 0 || ((SPIx)->SR & SPI_SR_BSY))

/**
 * @brief  SPI wait till end
 */
#define SPI_WAIT(SPIx)            while (SPI_IS_BUSY(SPIx))

/**
 * @brief  Checks if SPI is enabled
 */
#define SPI_CHECK_ENABLED(SPIx)   if (!((SPIx)->CR1 & 
SPI_CR1_SPE)) {return;}
```

```
/**
 * @brief  Checks if SPI is enabled and returns value from function if not
 */
#define SPI_CHECK_ENABLED_RESP(SPIx, val)   if (!((SPIx)->CR1 & SPI_CR1_SPE)) {return (val);}

/**
 * @}
 */

/**
 * @defgroup TM_SPI_Functions
 * @brief    Library Functions
 * @{
 */

/**
 * @brief  Initializes SPIx peripheral with custom pinspack and default other settings
 * @param  *SPIx: Pointer to SPIx peripheral you will use, where x is between 1 to 6
 * @param  pinspack: Pinspack you will use from default SPI table. This parameter can be a value of @ref TM_SPI_PinsPack_t enumeration
 * @retval None
 */
void TM_SPI_Init(SPI_TypeDef* SPIx, TM_SPI_PinsPack_t pinspack);

/**
 * @brief  Initializes SPIx peripheral with custom pinspack and SPI mode and default other settings
 * @param  *SPIx: Pointer to SPIx peripheral you will use, where x is between 1 to 6
 * @param  pinspack: Pinspack you will use from default SPI table. This parameter can be a value of @ref TM_SPI_PinsPack_t enumeration
 * @param  SPI_Mode: SPI mode you will use. This parameter can be a value of @ref TM_SPI_Mode_t enumeration
 * @retval None
 */
void TM_SPI_InitWithMode(SPI_TypeDef* SPIx, TM_SPI_PinsPack_t pinspack, TM_SPI_Mode_t SPI_Mode);

/**
 * @brief  Initializes SPIx peripheral with custom settings
 * @param  *SPIx: Pointer to SPIx peripheral you will use, where x is between 1 to 6
 * @param  pinspack: Pinspack you will use from default SPI table. This parameter can be a value of @ref TM_SPI_PinsPack_t enumeration
```

```
     * @param  SPI_BaudRatePrescaler: SPI baudrate prescaler. This
parameter can be a value of @ref SPI_BaudRatePrescaler
     * @param  SPI_Mode_t: SPI mode you will use. This parameter can
be a value of @ref TM_SPI_Mode_t enumeration
     * @param  SPI_Mode: SPI mode you will use:
     *           - SPI_Mode_Master: SPI in master mode (default)
     *           - SPI_Mode_Slave: SPI in slave mode
     * @param  SPI_FirstBit: select first bit for SPI
     *           - SPI_FirstBit_MSB: MSB is first bit (default)
     *           - SPI_FirstBit_LSB: LSB is first bit
     * @retval None
     */
    void TM_SPI_InitFull(SPI_TypeDef* SPIx, TM_SPI_PinsPack_t
pinspack, uint16_t SPI_BaudRatePrescaler, TM_SPI_Mode_t SPI_Mode_t,
uint16_t SPI_Mode, uint16_t SPI_FirstBit);

    /**
     * @brief  Calculates bits for SPI prescaler register to get
minimal prescaler value for SPI peripheral
     * @note   SPI has 8 prescalers available, 2,4,6,...,128,256
     * @note   This function will return you a bits you must set in
your CR1 register.
     *
     * @note   Imagine, you can use 20MHz max clock in your system,
your system is running on 168MHz, and you use SPI on APB2 bus.
     *         On 168 and 180MHz devices, APB2 works on Fclk/2, so 84
and 90MHz.
     *         So, if you calculate this, prescaler will need to be
84MHz / 20MHz = 4.xx, but if you use 4 prescaler, then you will be
over 20MHz.
     *         You need 8 prescaler then. This function will
calculate this.
     * @param  *SPIx: Pointer to SPIx peripheral you will use, where
x is between 1 to 6.
     *           Different SPIx works on different clock and is
important to know for which SPI you need prescaler.
     * @param  MAX_SPI_Frequency: Max SPI frequency you can use.
Function will calculate the minimum prescaler you need for that.
     *
     * @retval Bits combination for SPI_CR1 register, with aligned
location already, prepared to set parameter for @ref TM_SPI_InitFull()
function.
     */
    uint16_t TM_SPI_GetPrescalerFromMaxFrequency(SPI_TypeDef* SPIx,
uint32_t MAX_SPI_Frequency);

    /**
     * @brief  Sets data size for SPI at runtime
     * @note   You can select either 8 or 16 bits data array.
     * @param  *SPIx: Pointer to SPIx peripheral where data size will
be set
```

```
 * @param  DataSize: Datasize which will be used. This parameter
can be a value of @ref TM_SPI_DataSize_t enumeration
 * @retval Status of data size before changes happen
 */
TM_SPI_DataSize_t TM_SPI_SetDataSize(SPI_TypeDef* SPIx,
TM_SPI_DataSize_t DataSize);

/**
 * @brief  Sends single byte over SPI
 * @param  *SPIx: Pointer to SPIx peripheral you will use, where
x is between 1 to 6
 * @param  data: 8-bit data size to send over SPI
 * @retval Received byte from slave device
 */
static __INLINE uint8_t TM_SPI_Send(SPI_TypeDef* SPIx, uint8_t
data) {
	/* Check if SPI is enabled */
	SPI_CHECK_ENABLED_RESP(SPIx, 0);

	/* Wait for previous transmissions to complete if DMA TX
enabled for SPI */
	SPI_WAIT(SPIx);

	/* Fill output buffer with data */
	SPIx->DR = data;

	/* Wait for transmission to complete */
	SPI_WAIT(SPIx);

	/* Return data from buffer */
	return SPIx->DR;
}

/**
 * @brief  Sends and receives multiple bytes over SPIx
 * @param  *SPIx: Pointer to SPIx peripheral you will use, where
x is between 1 to 6
 * @param  *dataOut: Pointer to array with data to send over SPI
 * @param  *dataIn: Pointer to array to to save incoming data
 * @param  count: Number of bytes to send/receive over SPI
 * @retval None
 */
void TM_SPI_SendMulti(SPI_TypeDef* SPIx, uint8_t* dataOut,
uint8_t* dataIn, uint32_t count);

/**
 * @brief  Writes multiple bytes over SPI
 * @param  *SPIx: Pointer to SPIx peripheral you will use, where
x is between 1 to 6
 * @param  *dataOut: Pointer to array with data to send over SPI
 * @param  count: Number of elements to send over SPI
 * @retval None
```

```
     */
    void TM_SPI_WriteMulti(SPI_TypeDef* SPIx, uint8_t* dataOut, 
uint32_t count);

    /**
     * @brief  Receives multiple data bytes over SPI
     * @note   Selected SPI must be set in 16-bit mode
     * @param  *SPIx: Pointer to SPIx peripheral you will use, where 
x is between 1 to 6
     * @param  *dataIn: Pointer to 8-bit array to save data into
     * @param  dummy: Dummy byte  to be sent over SPI, to receive 
data back. In most cases 0x00 or 0xFF
     * @param  count: Number of bytes you want read from device
     * @retval None
     */
    void TM_SPI_ReadMulti(SPI_TypeDef* SPIx, uint8_t *dataIn, uint8_t 
dummy, uint32_t count);

    /**
     * @brief  Sends single byte over SPI
     * @note   Selected SPI must be set in 16-bit mode
     * @param  *SPIx: Pointer to SPIx peripheral you will use, where 
x is between 1 to 6
     * @param  data: 16-bit data size to send over SPI
     * @retval Received 16-bit value from slave device
     */
    static __INLINE uint16_t TM_SPI_Send16(SPI_TypeDef* SPIx, uint8_t 
data) {
        /* Check if SPI is enabled */
        SPI_CHECK_ENABLED_RESP(SPIx, 0);

        /* Wait for previous transmissions to complete if DMA TX 
enabled for SPI */
        SPI_WAIT(SPIx);

        /* Fill output buffer with data */
        SPIx->DR = data;

        /* Wait for transmission to complete */
        SPI_WAIT(SPIx);

        /* Return data from buffer */
        return SPIx->DR;
    }

    /**
     * @brief  Sends and receives multiple bytes over SPIx in 16-bit 
SPI mode
     * @note   Selected SPI must be set in 16-bit mode
     * @param  *SPIx: Pointer to SPIx peripheral you will use, where 
x is between 1 to 6
     * @param  *dataOut: Pointer to array with data to send over SPI
```

```
       * @param  *dataIn: Pointer to array to to save incoming data
       * @param  count: Number of 16-bit values to send/receive over
SPI
       * @retval None
       */
      void TM_SPI_SendMulti16(SPI_TypeDef* SPIx, uint16_t* dataOut,
uint16_t* dataIn, uint32_t count);

      /**
       * @brief  Writes multiple data via SPI in 16-bit SPI mode
       * @note   Selected SPI must be set in 16-bit mode
       * @param  *SPIx: Pointer to SPIx peripheral you will use, where
x is between 1 to 6
       * @param  *dataOut: Pointer to 16-bit array with data to send
over SPI
       * @param  count: Number of elements to send over SPI
       * @retval None
       */
      void TM_SPI_WriteMulti16(SPI_TypeDef* SPIx, uint16_t* dataOut,
uint32_t count);

      /**
       * @brief  Receives multiple data bytes over SPI in 16-bit SPI
mode
       * @note   Selected SPI must be set in 16-bit mode
       * @param  *SPIx: Pointer to SPIx peripheral you will use, where
x is between 1 to 6
       * @param  *dataIn: Pointer to 16-bit array to save data into
       * @param  dummy: Dummy 16-bit value to be sent over SPI, to
receive data back. In most cases 0x00 or 0xFF
       * @param  count: Number of 16-bit values you want read from
device
       * @retval None
       */
      void TM_SPI_ReadMulti16(SPI_TypeDef* SPIx, uint16_t* dataIn,
uint16_t dummy, uint32_t count);

      /**
       * @brief  Init custom SPI pins for your SPIx. This is callback
function and will be called from my library if needed.
       * @note   When you call TM_SPI_Init() function, and if you pass
TM_SPI_PinsPack_Custom to function,
       *         then this function will be called where you can
initialize custom pins for SPI peripheral
       *
       * @note   You have to initialize MOSI, MISO and SCK pin
       *
       * @param  *SPIx: Pointer to SPIx peripheral for which you have
to set your custom pin settings
       * @param  AlternateFunction: Alternate function which should be
used for GPIO initialization
       * @retval None
```

```
 * @note   With __weak parameter to prevent link errors if not defined by user
 */
void TM_SPI_InitCustomPinsCallback(SPI_TypeDef* SPIx, uint16_t AlternateFunction);

/**
 * @}
 */

/**
 * @}
 */

/**
 * @}
 */

/* C++ detection */
#ifdef __cplusplus
}
#endif

#endif
```

A.2.20. tm_stm32f4_usart

```
/**
 * @author  Tilen Majerle
 * @email   tilen@majerle.eu
 * @website http://stm32f4-discovery.com
 * @link    http://stm32f4-discovery.com/2014/04/library-04-connect-stm32f429-discovery-to-computer-with-usart/
 * @version v2.5
 * @ide     Keil uVision
 * @license GNU GPL v3
 * @brief   USART Library for STM32F4 with receive interrupt
 *
@verbatim
   ----------------------------------------------------------------------
    Copyright (C) Tilen Majerle, 2015

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    any later version.
```

```
         This program is distributed in the hope that it will be
useful,
         but WITHOUT ANY WARRANTY; without even the implied warranty
of
         MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
         GNU General Public License for more details.

         You should have received a copy of the GNU General Public
License
         along with this program.  If not, see
<http://www.gnu.org/licenses/>.
        ----------------------------------------------------------------
--------
     @endverbatim
      */
     #ifndef TM_USART_H
     #define TM_USART_H 250

     /* C++ detection */
     #ifdef __cplusplus
     extern "C" {
     #endif

     /**
      * @addtogroup TM_STM32F4xx_Libraries
      * @{
      */

     /**
      * @defgroup TM_USART
      * @brief    TM USART Library for STM32F4xx - http://stm32f4-
discovery.com/2014/04/library-04-connect-stm32f429-discovery-to-
computer-with-usart/
      * @{
      *
      * <b>Library works for all 8 U(S)ARTs which are supported on
STM32F4xx devices.</b>
      *
      * \par USART receive interrupt handlers
      *
      * Every USART channel has it's own receive interrupt which
stores incoming data into cyclic buffer.
      * If you want to use your own receive handler, then you have to
open defines.h files and set a define.
     @verbatim
     //Use custom IRQ Receive handler
     //Change X with possible U(S)ARTs: USART1, USART2, USART3, UART4,
UART5, USART6, UART7, UART8
     #define TM_X_USE_CUSTOM_IRQ
     @endverbatim
      * After you set define, you have to create a function, which
will handle custom request
```

```
      @verbatim
      //Change X with possible U(S)ARTs: USART1, USART2, USART3, UART4,
UART5, USART6, UART7, UART8
      //Parameter c is a received character
      void TM_X_ReceiveHandler(uint8_t c) {
         //Do your stuff here when byte is received
      }
      @endverbatim
       * @note If you use custom receive interrupt handler, then
incoming data is not stored in internal buffer
       *
       * \par USART Internal cyclic buffer
       *
       * In your project you can set internal cyclic buffer length,
default is 32Bytes, with:
      @verbatim
      //Set buffer sie for all buffers
      #define USART_BUFFER_SIZE number_of_bytes
      @endverbatim
       * in your project's defines.h file. This will set default length
for each buffer.
       * So if you are working with F429 (it has 8 U(S)ARTs) then you
will use 8kB RAM if
       * you set define above to 1024.
       *
       * As of version 2.0, you can now set different buffer sizes for
different U(S)ARTs.
       * If you don't change anything, then all USART's have buffer
length of value, stored in
       * <code>USART_BUFFER_SIZE</code> define. If you want let's say
just for USART1 to be 1kB, but others default value,
       * you can add define below in defines.h file:
      @verbatim
      //Buffer length for USART1 is 1kB, for others is still
TM_USART_BUFFER_SIZE
      #define TM_USART1_BUFFER_SIZE 1024
      @endverbatim
       *
       * Other possible settings are (for other U(S)ARTs):
       *   - TM_USART1_BUFFER_SIZE
       *   - TM_USART2_BUFFER_SIZE
       *   - TM_USART3_BUFFER_SIZE
       *   - TM_UART4_BUFFER_SIZE
       *   - TM_UART5_BUFFER_SIZE
       *   - TM_USART6_BUFFER_SIZE
       *   - TM_UART7_BUFFER_SIZE
       *   - TM_UART8_BUFFER_SIZE
       *
       * \par Custom string delimiter for @ref TM_USART_Gets() function
       *
       * As of version 2.5, you can now set custom string delimiter for
@ref TM_USART_Gets() function.
```

```
      * By default, LF (Line Feed) character was used, but now you can
select custom character using @ref
TM_USART_SetCustomStringEndCharacter() function.
      *
      * \par Pinout
      *
     @verbatim
                 |PINSPACK 1     |PINSPACK 2     |PINSPACK 3
     U(S)ARTX    |TX     RX      |TX     RX      |TX     RX

     USART1      |PA9    PA10    |PB6    PB7     |-      -
     USART2      |PA2    PA3     |PD5    PD6     |-      -
     USART3      |PB10   PB11    |PC10   PC11    |PD8    PD9
     UART4       |PA0    PA1     |PC10   PC11    |-      -
     UART5       |PC12   PD2     |-      -       |-      -
     USART6      |PC6    PC7     |PG14   PG9     |-      -
     UART7       |PE8    PE7     |PF7    PF6     |-      -
     UART8       |PE1    PE0     |-      -       |-      -
     @endverbatim
      *
      * In case these pins are not good for you, you can use
      * TM_USART_PinsPack_Custom in function and callback function
will be called,
      * where you can initialize your custom pinout for your USART
peripheral
      *
      * \par Change USART default operation modes
      *
      * In this section, you can change USART functionality.
      * Do this only in case you know what are you doing!
      *
      * Open \ref defines.h file, copy define you want to change and
fill settings
     @verbatim
     //Change X with possible U(S)ARTs: USART1, USART2, USART3, UART4,
UART5, USART6, UART7, UART8
     //Set flow control
     #define TM_X_HARDWARE_FLOW_CONTROL
     TM_USART_HardwareFlowControl_None
     //Set mode
     #define TM_X_MODE                              USART_Mode_Tx |
USART_Mode_Rx
     //Set parity
     #define TM_X_PARITY                            USART_Parity_No
     //Set stopbits
     #define TM_X_STOP_BITS                         USART_StopBits_1
     //Set USART datasize
     #define TM_X_WORD_LENGTH                   USART_WordLength_8b
     @endverbatim
      *
      * \par Changelog
      *
```

```
      @verbatim
       Version 2.5
         - April 15, 2015
         - Added support for custom character for string delimiter

       Version 2.4
         - April 09, 2015
         - Added support for new function
TM_USART_InitWithFlowControl()

       Version 2.3.2
         - March 21, 2015
         - Code optimizations

       Version 2.3.2
         - March 17, 2015
         - Added support for Doxygen

       Version 2.3
         - March 14, 2015
         - Added support for STM32F446xx devices
         - Changed function name for custom pins initialization
callback

       Version 2.2
         - March 10, 2015
         - Updated to be more independent of STD/HAL drivers but still
not totally

       Version 2.1
         - March 08, 2015
         - Output pins are more clear initialized.
         - TM GPIO library is now required to get USART to work
properly

       Version 2.0
         - December 21, 2014
         - New cyclic buffer system,
            each U(S)ART can have different buffer size (less RAM can
be used for USART purpose)
         - Added function to check if buffer is full,
         - TM_USART_Gets now returns 0 till '\n' is not available in
buffer or buffer is full
            Useful for prevent infinite loop if '\n' never happen

       Version 1.0
         - First release
      @endverbatim
       *
       * \b Dependencies
       *
      @verbatim
```

```
 - STM32F4xx
 - STM32F4xx RCC
 - STM32F4xx GPIO
 - STM32F4xx USART
 - attributes.h
 - defines.h
 - TM GPIO
@endverbatim
 */
#include "misc.h"
#include "stm32f4xx.h"
#include "stm32f4xx_rcc.h"
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_usart.h"
#include "attributes.h"
#include "defines.h"
#include "tm_stm32f4_gpio.h"

/* F405/407/415/417/F446 */
#if defined (STM32F40_41xxx) || defined(STM32F446xx)
#define USE_USART1
#define USE_USART2
#define USE_USART3
#define USE_UART4
#define USE_UART5
#define USE_USART6
#endif
/* F427/429/437/439 */
#if defined (STM32F427_437xx) || defined (STM32F429_439xx)
#define USE_USART1
#define USE_USART2
#define USE_USART3
#define USE_UART4
#define USE_UART5
#define USE_USART6
#define USE_UART7
#define USE_UART8
#endif
/* F401/411 */
#if defined (STM32F401xx) || defined(STM32F411xE)
#define USE_USART1
#define USE_USART2
#define USE_USART6
#endif


/**
 * @defgroup TM_USART_Typedefs
 * @brief    USART Typedefs
 * @{
 */
```

```
/**
 * @brief  USART PinsPack enumeration to select pins combination
for USART
 */
typedef enum {
	TM_USART_PinsPack_1,     /*!< Select PinsPack1 from Pinout
table for specific USART */
	TM_USART_PinsPack_2,     /*!< Select PinsPack2 from Pinout
table for specific USART */
	TM_USART_PinsPack_3,     /*!< Select PinsPack3 from Pinout
table for specific USART */
	TM_USART_PinsPack_Custom /*!< Select custom pins for
specific USART, callback will be called, look @ref
TM_USART_InitCustomPinsCallback */
} TM_USART_PinsPack_t;

/**
 * @brief  USART Hardware flow control selection
 * @note   Corresponsing pins must be initialized in case you
don't use "None" options
 */
typedef enum {
	TM_USART_HardwareFlowControl_None = 0x0000,   /*!< No flow
control */
	TM_USART_HardwareFlowControl_RTS = 0x0100,    /*!< RTS flow
control */
	TM_USART_HardwareFlowControl_CTS = 0x0200,    /*!< CTS flow
control */
	TM_USART_HardwareFlowControl_RTS_CTS = 0x0300 /*!< RTS and
CTS flow control */
} TM_USART_HardwareFlowControl_t;

/**
 * @}
 */

/**
 * @defgroup TM_USART_Macros
 * @brief    USART default values for defines
 * @{
 *
 * All values can be overwritten in your project's defines.h
file.
 *
 * Do this only in case you know what are you doing.
 */

/* Default buffer size for each USART */
#ifndef USART_BUFFER_SIZE
#define USART_BUFFER_SIZE                 32
#endif
```

```
/* Set default buffer size for specific USART if not set by user
*/
#ifndef TM_USART1_BUFFER_SIZE
#define TM_USART1_BUFFER_SIZE              USART_BUFFER_SIZE
#endif
#ifndef TM_USART2_BUFFER_SIZE
#define TM_USART2_BUFFER_SIZE              USART_BUFFER_SIZE
#endif
#ifndef TM_USART3_BUFFER_SIZE
#define TM_USART3_BUFFER_SIZE              USART_BUFFER_SIZE
#endif
#ifndef TM_UART4_BUFFER_SIZE
#define TM_UART4_BUFFER_SIZE               USART_BUFFER_SIZE
#endif
#ifndef TM_UART5_BUFFER_SIZE
#define TM_UART5_BUFFER_SIZE               USART_BUFFER_SIZE
#endif
#ifndef TM_USART6_BUFFER_SIZE
#define TM_USART6_BUFFER_SIZE              USART_BUFFER_SIZE
#endif
#ifndef TM_UART7_BUFFER_SIZE
#define TM_UART7_BUFFER_SIZE               USART_BUFFER_SIZE
#endif
#ifndef TM_UART8_BUFFER_SIZE
#define TM_UART8_BUFFER_SIZE               USART_BUFFER_SIZE
#endif

/* NVIC Global Priority */
#ifndef USART_NVIC_PRIORITY
#define USART_NVIC_PRIORITY                0x06
#endif

/* U(S)ART settings, can be changed in your defines.h project
file */
/* USART1 default settings */
#ifndef TM_USART1_HARDWARE_FLOW_CONTROL
#define TM_USART1_HARDWARE_FLOW_CONTROL
TM_USART_HardwareFlowControl_None
#endif
#ifndef TM_USART1_MODE
#define TM_USART1_MODE
USART_Mode_Tx | USART_Mode_Rx
#endif
#ifndef TM_USART1_PARITY
#define TM_USART1_PARITY                        USART_Parity_No
#endif
#ifndef TM_USART1_STOP_BITS
#define TM_USART1_STOP_BITS                     USART_StopBits_1
#endif
#ifndef TM_USART1_WORD_LENGTH
#define TM_USART1_WORD_LENGTH
USART_WordLength_8b
```

```
#endif

/* USART2 default settings */
#ifndef TM_USART2_HARDWARE_FLOW_CONTROL
#define TM_USART2_HARDWARE_FLOW_CONTROL
TM_USART_HardwareFlowControl_None
#endif
#ifndef TM_USART2_MODE
#define TM_USART2_MODE
USART_Mode_Tx | USART_Mode_Rx
#endif
#ifndef TM_USART2_PARITY
#define TM_USART2_PARITY                        USART_Parity_No
#endif
#ifndef TM_USART2_STOP_BITS
#define TM_USART2_STOP_BITS                     USART_StopBits_1
#endif
#ifndef TM_USART2_WORD_LENGTH
#define TM_USART2_WORD_LENGTH
USART_WordLength_8b
#endif

/* USART3 default settings */
#ifndef TM_USART3_HARDWARE_FLOW_CONTROL
#define TM_USART3_HARDWARE_FLOW_CONTROL
TM_USART_HardwareFlowControl_None
#endif
#ifndef TM_USART3_MODE
#define TM_USART3_MODE
USART_Mode_Tx | USART_Mode_Rx
#endif
#ifndef TM_USART3_PARITY
#define TM_USART3_PARITY                        USART_Parity_No
#endif
#ifndef TM_USART3_STOP_BITS
#define TM_USART3_STOP_BITS                     USART_StopBits_1
#endif
#ifndef TM_USART3_WORD_LENGTH
#define TM_USART3_WORD_LENGTH
USART_WordLength_8b
#endif

/* UART4 default settings */
#ifndef TM_UART4_HARDWARE_FLOW_CONTROL
#define TM_UART4_HARDWARE_FLOW_CONTROL
TM_USART_HardwareFlowControl_None
#endif
#ifndef TM_UART4_MODE
#define TM_UART4_MODE                           USART_Mode_Tx |
USART_Mode_Rx
#endif
#ifndef TM_UART4_PARITY
```

```
#define TM_UART4_PARITY
USART_Parity_No
#endif
#ifndef TM_UART4_STOP_BITS
#define TM_UART4_STOP_BITS                              USART_StopBits_1
#endif
#ifndef TM_UART4_WORD_LENGTH
#define TM_UART4_WORD_LENGTH
USART_WordLength_8b
#endif

/* UART5 default settings */
#ifndef TM_UART5_HARDWARE_FLOW_CONTROL
#define TM_UART5_HARDWARE_FLOW_CONTROL
TM_USART_HardwareFlowControl_None
#endif
#ifndef TM_UART5_MODE
#define TM_UART5_MODE                                   USART_Mode_Tx |
USART_Mode_Rx
#endif
#ifndef TM_UART5_PARITY
#define TM_UART5_PARITY
USART_Parity_No
#endif
#ifndef TM_UART5_STOP_BITS
#define TM_UART5_STOP_BITS                              USART_StopBits_1
#endif
#ifndef TM_UART5_WORD_LENGTH
#define TM_UART5_WORD_LENGTH
USART_WordLength_8b
#endif

/* USART6 default settings */
#ifndef TM_USART6_HARDWARE_FLOW_CONTROL
#define TM_USART6_HARDWARE_FLOW_CONTROL
TM_USART_HardwareFlowControl_None
#endif
#ifndef TM_USART6_MODE
#define TM_USART6_MODE
USART_Mode_Tx | USART_Mode_Rx
#endif
#ifndef TM_USART6_PARITY
#define TM_USART6_PARITY                                USART_Parity_No
#endif
#ifndef TM_USART6_STOP_BITS
#define TM_USART6_STOP_BITS                             USART_StopBits_1
#endif
#ifndef TM_USART6_WORD_LENGTH
#define TM_USART6_WORD_LENGTH
USART_WordLength_8b
#endif
```

```
/* UART7 default settings */
#ifndef TM_UART7_HARDWARE_FLOW_CONTROL
#define TM_UART7_HARDWARE_FLOW_CONTROL
TM_USART_HardwareFlowControl_None
#endif
#ifndef TM_UART7_MODE
#define TM_UART7_MODE                                       USART_Mode_Tx |
USART_Mode_Rx
#endif
#ifndef TM_UART7_PARITY
#define TM_UART7_PARITY
USART_Parity_No
#endif
#ifndef TM_UART7_STOP_BITS
#define TM_UART7_STOP_BITS                                  USART_StopBits_1
#endif
#ifndef TM_UART7_WORD_LENGTH
#define TM_UART7_WORD_LENGTH
USART_WordLength_8b
#endif

/* UART8 default settings */
#ifndef TM_UART8_HARDWARE_FLOW_CONTROL
#define TM_UART8_HARDWARE_FLOW_CONTROL
TM_USART_HardwareFlowControl_None
#endif
#ifndef TM_UART8_MODE
#define TM_UART8_MODE                                       USART_Mode_Tx |
USART_Mode_Rx
#endif
#ifndef TM_UART8_PARITY
#define TM_UART8_PARITY
USART_Parity_No
#endif
#ifndef TM_UART8_STOP_BITS
#define TM_UART8_STOP_BITS                                  USART_StopBits_1
#endif
#ifndef TM_UART8_WORD_LENGTH
#define TM_UART8_WORD_LENGTH
USART_WordLength_8b
#endif

/**
 * @brief  Wait till USART finishes transmission
 */
#define USART_TXEMPTY(USARTx)               ((USARTx)->SR &
USART_FLAG_TXE)
#define USART_WAIT(USARTx)                  do { while
(!USART_TXEMPTY(USARTx)); } while (0)

/**
 * @brief  Default string delimiter for USART
```

```
 */
#define USART_STRING_DELIMITER              '\n'

 /**
 * @}
 */

/**
 * @defgroup TM_USART_Functions
 * @brief    USART Functions
 * @{
 */

/**
 * @brief  Initializes USARTx peripheral and corresponding pins
 * @param  *USARTx: Pointer to USARTx peripheral you will use
 * @param  pinspack: This parameter can be a value of @ref
TM_USART_PinsPack_t enumeration
 * @param  baudrate: Baudrate number for USART communication
 * @retval None
 */
void TM_USART_Init(USART_TypeDef* USARTx, TM_USART_PinsPack_t
pinspack, uint32_t baudrate);

/**
 * @brief  Initializes USARTx peripheral and corresponding pins
with custom hardware flow control mode
 * @note   Hardware flow control pins are not initialized. Easy
solution is to use @arg TM_USART_PinsPack_Custom pinspack option
 *         when you call @ref TM_USART_Init() function and
initialize all USART pins at a time inside @ref
TM_USART_InitCustomPinsCallback()
 *         callback function, which will be called from my
library
 * @param  *USARTx: Pointer to USARTx peripheral you will use
 * @param  pinspack: This parameter can be a value of @ref
TM_USART_PinsPack_t enumeration
 * @param  baudrate: Baudrate number for USART communication
 * @param  FlowControl: Flow control mode you will use. This
parameter can be a value of @ref TM_USART_HardwareFlowControl_t
enumeration
 * @retval None
 */
void TM_USART_InitWithFlowControl(USART_TypeDef* USARTx,
TM_USART_PinsPack_t pinspack, uint32_t baudrate,
TM_USART_HardwareFlowControl_t FlowControl);

/**
 * @brief  Puts character to USART port
 * @param  *USARTx: Pointer to USARTx peripheral you will use
 * @param  c: character to be send over USART
 * @retval None
```

```
 */
static __INLINE void TM_USART_Putc(USART_TypeDef* USARTx,
volatile char c) {
      /* Check USART if enabled */
      if ((USARTx->CR1 & USART_CR1_UE)) {
            /* Wait to be ready, buffer empty */
            USART_WAIT(USARTx);
            /* Send data */
            USARTx->DR = (uint16_t)(c & 0x01FF);
            /* Wait to be ready, buffer empty */
            USART_WAIT(USARTx);
      }
}

/**
 * @brief  Puts string to USART port
 * @param  *USARTx: Pointer to USARTx peripheral you will use
 * @param  *str: Pointer to string to send over USART
 * @retval None
 */
void TM_USART_Puts(USART_TypeDef* USARTx, char* str);

/**
 * @brief  Sends data array to USART port
 * @param  *USARTx: Pointer to USARTx peripheral you will use
 * @param  *DataArray: Pointer to data array to be sent over
USART
 * @param  count: Number of elements in data array to be send
over USART
 * @retval None
 */
void TM_USART_Send(USART_TypeDef* USARTx, uint8_t* DataArray,
uint16_t count);

/**
 * @brief  Gets character from internal USART buffer
 * @param  *USARTx: Pointer to USARTx peripheral you will use
 * @retval Character from buffer, or 0 if nothing in buffer
 */
uint8_t TM_USART_Getc(USART_TypeDef* USARTx);

/**
 * @brief  Gets string from USART
 *
 *          This function can create a string from USART received
data.
 *
 *          It generates string until "\n" is not recognized or
buffer length is full.
 *
 * @note   As of version 1.5, this function automatically adds
0x0A (Line feed) at the end of string.
```

```
       * @param  *USARTx: Pointer to USARTx peripheral you will use
       * @param  *buffer: Pointer to buffer where data will be stored
from buffer
       * @param  bufsize: maximal number of characters we can add to
your buffer, including leading zero
       * @retval Number of characters in buffer
       */
      uint16_t TM_USART_Gets(USART_TypeDef* USARTx, char* buffer,
uint16_t bufsize);

      /**
       * @brief  Checks if character c is available in internal buffer
       * @param  *USARTx: Pointer to USARTx peripheral you will use
       * @param  c: character to check if it is in USARTx's buffer
       * @retval Character status:
       *            - 0: Character was not found
       *            - > 0: Character has been found in buffer
       */
      uint8_t TM_USART_FindCharacter(USART_TypeDef* USARTx, uint8_t c);

      /**
       * @brief  Checks if internal USARTx buffer is empty
       * @param  *USARTx: Pointer to USARTx peripheral you will use
       * @retval Buffer empty status:
       *            - 0: Buffer is not empty
       *            - > 0: Buffer is empty
       */
      uint8_t TM_USART_BufferEmpty(USART_TypeDef* USARTx);

      /**
       * @brief  Checks if internal USARTx buffer is full
       * @param  *USARTx: Pointer to USARTx peripheral you will use
       * @retval Buffer full status:
       *            - 0: Buffer is not full
       *            - > 0: Buffer is full
       */
      uint8_t TM_USART_BufferFull(USART_TypeDef* USARTx);

      /**
       * @brief  Clears internal USART buffer
       * @param  *USARTx: Pointer to USARTx peripheral you will use
       * @retval None
       */
      void TM_USART_ClearBuffer(USART_TypeDef* USARTx);

      /**
       * @brief  Sets custom character for @ref TM_USART_Gets()
function to detect when string ends
       * @param  *USARTx: Pointer to USARTx peripheral you will use
       * @param  Character: Character value to be used as string end
       * @note   Character will also be added at the end for your
buffer when calling @ref TM_USART_Gets() function
```

```
 * @retval None
 */
void TM_USART_SetCustomStringEndCharacter(USART_TypeDef* USARTx,
uint8_t Character);

/**
 * @brief  Callback for custom pins initialization for USARTx.
 *
 *         When you call @ef TM_USART_Init() function, and if you
pass @arg TM_USART_PinsPack_Custom to function,
 *         then this function will be called where you can
initialize custom pins for USART peripheral.
 * @param  *USARTx: Pointer to USARTx peripheral you will use for
initialization
 * @param  AlternateFunction: Alternate function which should be
used for GPIO initialization
 * @retval None
 * @note   With __weak parameter to prevent link errors if not
defined by user
 */
void TM_USART_InitCustomPinsCallback(USART_TypeDef* USARTx,
uint16_t AlternateFunction);

/**
 * @brief  Callback function for receive interrupt on USART1 in
case you have enabled custom USART handler mode
 * @note   With __weak parameter to prevent link errors if not
defined by user
 * @param  c: character received via USART
 * @retval None
 */
__weak void TM_USART1_ReceiveHandler(uint8_t c);

/**
 * @brief  Callback function for receive interrupt on USART2 in
case you have enabled custom USART handler mode
 * @note   With __weak parameter to prevent link errors if not
defined by user
 * @param  c: character received via USART
 * @retval None
 */
__weak void TM_USART2_ReceiveHandler(uint8_t c);

/**
 * @brief  Callback function for receive interrupt on USART3 in
case you have enabled custom USART handler mode
 * @note   With __weak parameter to prevent link errors if not
defined by user
 * @param  c: character received via USART
 * @retval None
 */
__weak void TM_USART3_ReceiveHandler(uint8_t c);
```

```
/**
 * @brief  Callback function for receive interrupt on UART4 in
case you have enabled custom USART handler mode
 * @note   With __weak parameter to prevent link errors if not
defined by user
 * @param  c: character received via USART
 * @retval None
 */
__weak void TM_UART4_ReceiveHandler(uint8_t c);

/**
 * @brief  Callback function for receive interrupt on UART5 in
case you have enabled custom USART handler mode
 * @note   With __weak parameter to prevent link errors if not
defined by user
 * @param  c: character received via USART
 * @retval None
 */
__weak void TM_UART5_ReceiveHandler(uint8_t c);

/**
 * @brief  Callback function for receive interrupt on USART6 in
case you have enabled custom USART handler mode
 * @note   With __weak parameter to prevent link errors if not
defined by user
 * @param  c: character received via USART
 * @retval None
 */
__weak void TM_USART6_ReceiveHandler(uint8_t c);

/**
 * @brief  Callback function for receive interrupt on UART7 in
case you have enabled custom USART handler mode
 * @note   With __weak parameter to prevent link errors if not
defined by user
 * @param  c: character received via USART
 * @retval None
 */
__weak void TM_UART7_ReceiveHandler(uint8_t c);

/**
 * @brief  Callback function for receive interrupt on UART8 in
case you have enabled custom USART handler mode
 * @note   With __weak parameter to prevent link errors if not
defined by user
 * @param  c: character received via USART
 * @retval None
 */
__weak void TM_UART8_ReceiveHandler(uint8_t c);

/**
```

```
 * @}
 */

/**
 * @}
 */

/**
 * @}
 */

/* C++ detection */
#ifdef __cplusplus
}
#endif

#endif
```

A.2.21. tm_stm32f4_vcp

```
/**
 * @author  Tilen Majerle
 * @email   tilen@majerle.eu
 * @website http://stm32f4-discovery.com
 * @link    http://stm32f4-discovery.com/2014/08/library-24-
virtual-com-port-vcp-stm32f4xx/
 * @version v1.2
 * @ide     Keil uVision
 * @license GNU GPL v3
 * @brief   USB Virtual COM Port for STM32F4xx devices
 *
@verbatim
   ----------------------------------------------------------------------
--------
   Copyright (C) Tilen Majerle, 2015

   This program is free software: you can redistribute it and/or
modify
   it under the terms of the GNU General Public License as
published by
   the Free Software Foundation, either version 3 of the
License, or
   any later version.

   This program is distributed in the hope that it will be
useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty
of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.
```

```
         You should have received a copy of the GNU General Public
License
         along with this program.  If not, see
<http://www.gnu.org/licenses/>.
        ----------------------------------------------------------------------
--------
      @endverbatim
       */
      #ifndef TM_USB_VCP_H
      #define TM_USB_VCP_H   120

      /* C++ detection */
      #ifdef __cplusplus
      extern "C" {
      #endif

      /**
       * @addtogroup TM_STM32F4xx_Libraries
       * @{
       */

      /**
       * @defgroup TM_USB_VCP
       * @brief    USB Virtual COM Port for STM32F4xx devices -
http://stm32f4-discovery.com/2014/08/library-24-virtual-com-port-vcp-
stm32f4xx/
       * @{
       *
       * With this library, your STM32F4xx will be seen to your
computer as Virtual COM Port (VCP).
       * To be able to work, you have to install ST's VCP Driver, from
link below:
       *
       *         http://www.st.com/web/en/catalog/tools/PF257938
       *
       * This library can work in 2 ways.
       * First and default is Full-Speed mode, second option is High-
Speed mode.
       * Also, different modes have different pinouts.
       *
       * In default settings, USB FS mode is selected.
       *
       * STM32F4-Discovery has USB connected to FS mode, but
       * STM32F429-Discovery has connected it to USB HS in FS mode.
       * But if you have cable, like me, USB->4wires, you can connect
Data+ and Data- to any pin on Discovery board.
       * I did this, to check, if both mdoes work on bots discovery
boards and yes, it worked.
       * For security reasons set 22Ohm resistors in serial to your
data pins.
       *
       * USB FS MODE (micro USB connected on STM32F4 Discovery board)
```

```
       *    - This is default option and don't need any special
settings.
       *
       * \par Pinout for USB FS mode
       *
      @verbatim
      USB              STM32F4xx
      Data +           PA12
      Data -           PA11
      @endverbatim
       *
       * USB HS in FS mode (micro USB connected on STM32F429 Discovery
board)
       *
       *    If you are working with STM32F429 Discovery board, and you
want to use microUSB connector for VCP,
       *    then set define below in your defines.h file
       *
      @verbatim
      //Activate USB HS in FS mode
      #define USE_USB_OTG_HS
      @endverbatim
       *
       * \par Pinout for USB HS in FS mode
       *
      @verbatim
      USB              STM32F4xx
      Data +           PB15
      Data -           PB14
      @endverbatim
       *
       *
       * \par Changelog
       *
      @verbatim
       Version 1.2
        - March 08, 2015
        - Added options to get user settings from terminal
        - Baudrate, stop bits, parity, data bits.
        - Useful if you make USB->UART converter like FTDI

       Version 1.1
        - December 27, 2014
        - Added advanced functions for string operations
        - Now, Gets function will wait till buffer is full or \n is
received
        - This is prevent for while loop if \n character is not
received

       Version 1.0
        - First release
      @endverbatim
```

```
 *
 * \par Dependencies
 *
@verbatim
 - STM32F4xx
 - STM32F4xx RCC
 - STM32F4xx GPIO
 - STM32F4xx EXTI
 - misc.h
 - defines.h
 - USB CDC DEVICE
@endverbatim
 */

#include "stm32f4xx.h"
#include "stm32f4xx_rcc.h"
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_exti.h"
#include "misc.h"
#include "defines.h"

/* Parts of USB device */
#include "usb_vcp/usbd_cdc_core.h"
#include "usb_vcp/usb_conf.h"
#include "usb_vcp/usbd_desc.h"
#include "usb_vcp/usbd_cdc_vcp.h"

/**
 * @defgroup TM_USB_VCP_Macros
 * @brief    Library defines
 * @{
 */

/**
 * @brief  Default buffer length
 * @note   Increase this value if you need more memory for VCP
receive data
 */
#ifndef USB_VCP_RECEIVE_BUFFER_LENGTH
#define USB_VCP_RECEIVE_BUFFER_LENGTH       128
#endif

/**
 * @}
 */

/**
 * @defgroup TM_USB_VCP_Typedefs
 * @brief    Library Typedefs
 * @{
 */
```

```
    /**
     * @brief VCP Result Enumerations
     */
    typedef enum {
        TM_USB_VCP_OK,                      /*!< Everything ok */
        TM_USB_VCP_ERROR,                   /*!< An error occurred */
        TM_USB_VCP_RECEIVE_BUFFER_FULL, /*!< Receive buffer is full
*/
        TM_USB_VCP_DATA_OK,                 /*!< Data OK */
        TM_USB_VCP_DATA_EMPTY,              /*!< Data empty */
        TM_USB_VCP_NOT_CONNECTED,           /*!< Not connected to PC */
        TM_USB_VCP_CONNECTED,               /*!< Connected to PC */
        TM_USB_VCP_DEVICE_SUSPENDED,        /*!< Device is suspended */
        TM_USB_VCP_DEVICE_RESUMED           /*!< Device is resumed */
    } TM_USB_VCP_Result;

    /**
     * @brief  Structure for USART if you are working USB/UART
converter with STM32F4xx
     */
    typedef struct {
        uint32_t Baudrate; /*!< Baudrate, which is set by user on
terminal.
                                 Value is number of bits per second,
for example: 115200 */
        uint8_t Stopbits;  /*!< Stop bits, which is set by user on
terminal.
                                 Possible values:
                                  - 0: 1 stop bit
                                  - 1: 1.5 stop bits
                                  - 2: 2 stop bits */
        uint8_t DataBits;  /*!< Data bits, which is set by user on
terminal.
                                Possible values:
                                  - 5: 5 data bits
                                  - 6: 6 data bits
                                  - 7: 7 data bits
                                  - 8: 8 data bits
                                  - 9: 9 data bits */
        uint8_t Parity;    /*!< Parity, which is set by user on
terminal.
                                Possible values:
                                  - 0: No parity
                                  - 1: Odd parity
                                  - 2: Even parity
                                  - 3: Mark parity
                                  - 4: Space parity */
        uint8_t Changed;   /*!< When you check for settings in my
function,
                                 this will be set to 1 if user has
changed parameters,
```

```
                                 so you can reinitialize USART
peripheral if you need to. */
      } TM_USB_VCP_Settings_t;

      /**
       * @}
       */

      /**
       * @defgroup TM_USB_VCP_Functions
       * @brief    Library Functions
       * @{
       */

      /**
       * @brief  Initializes USB VCP
       * @param  None
       * @retval TM_USB_VCP_OK
       */
      TM_USB_VCP_Result TM_USB_VCP_Init(void);

      /**
       * @brief  Reads settings from user
       * @note   These settings are set in terminal on PC
       * @param  *Settings: Pointer to TM_USB_VCP_Settings_t structure
where to save data
       * @retval TM_USB_VCP_OK
       */
      TM_USB_VCP_Result TM_USB_VCP_GetSettings(TM_USB_VCP_Settings_t*
Settings);

      /**
       * @brief  Gets received character from internal buffer
       * @param  *c: pointer to store new character to
       * @retval Character status:
       *            - TM_USB_VCP_DATA_OK: Character is valid inside
*c_str
       *            - TM_USB_VCP_DATA_EMPTY: No character in *c
       */
      TM_USB_VCP_Result TM_USB_VCP_Getc(uint8_t* c);

      /**
       * @brief  Puts character to USB VCP
       * @param  c: character to send over USB
       * @retval TM_USB_VCP_OK
       */
      TM_USB_VCP_Result TM_USB_VCP_Putc(volatile char c);

      /**
       * @brief  Gets string from VCP port
       *
```

```
     * @note   To use this method, you have to send \n (0x0D) at the
end of your string,
     *        otherwise data can be lost and you will fall in
infinite loop.
     * @param  *buffer: Pointer to buffer variable where to save
string
     * @param  bufsize: Maximum buffer size
     * @retval Number of characters in buffer:
     *           - 0: String not valid
     *           - > 0: String valid, number of characters inside
string
     */
    uint16_t TM_USB_VCP_Gets(char* buffer, uint16_t bufsize);

    /**
     * @brief  Puts string to USB VCP
     * @param  *str: Pointer to string variable
     * @retval TM_USB_VCP_OK
     */
    TM_USB_VCP_Result TM_USB_VCP_Puts(char* str);

    /**
     * @brief  Sends array of data to USB VCP
     * @param  *DataArray: Pointer to 8-bit data array to be sent
over USB
     * @param  Length: Number of elements to sent in units of bytes
     * @retval Sending status
     */
    TM_USB_VCP_Result TM_USB_VCP_Send(uint8_t* DataArray, uint32_t
Length);

    /**
     * @brief  Gets VCP status
     * @param  None
     * @retval Device status:
     *           - TM_USB_VCP_CONNECTED: Connected to computer
     *           - other: Not connected and not ready to communicate
     */
    TM_USB_VCP_Result TM_USB_VCP_GetStatus(void);

    /**
     * @brief  Checks if receive buffer is empty
     * @param  None
     * @retval Buffer status:
     *           - 0: Buffer is not empty
     *           - > 0: Buffer is empty
     */
    uint8_t TM_USB_VCP_BufferEmpty(void);

    /**
     * @brief  Checks if receive buffer is fukk
     * @param  None
```

```
 * @retval Buffer status:
 *            - 0: Buffer is not full
 *            - > 0: Buffer is full
 */
uint8_t TM_USB_VCP_BufferFull(void);

/**
 * @brief  Checks if character is in buffer
 * @param  c: Character to be checked if available in buffer
 * @retval Character status:
 *            - 0: Character is not in buffer
 *            - > 0: Character is in buffer
 */
uint8_t TM_USB_VCP_FindCharacter(volatile char c);

/* Internal functions */
extern TM_USB_VCP_Result TM_INT_USB_VCP_AddReceived(uint8_t c);

/**
 * @}
 */

/**
 * @}
 */

/**
 * @}
 */

/* C++ detection */
#ifdef __cplusplus
}
#endif

#endif
```

A.3. C Files for Experimental System

A.3.1. Main.c

```
#include <tm_stm32f4_usb_vcp.h>
#include <tm_stm32f4_disco.h>
#include <tm_stm32f4_delay.h>

#define HIGH 1
#define LOW  0

// simple packet protocol stuff
#include <packet/SimplifiedPacketProtocol.h>

#include "RTC_Initialization.h"
#include "pdm_filter.h"
#include "DC_Motor_Initializations.h"
#include "DC_Motor_Actuation.h"
#include "DC_Motor_PID.h"
#include "Stepper_Initializations.h"
#include "Stepper_Actuation.h"
#include "Stepper_Motor_Actuation.h"
#include "ADC__Initialization.h"
#include "ADC_Measurement.h"
#include "ENC_Initialization.h"
#include "ENC_measurement.h"
#include "IMU_Initialization.h"
#include "IMU_measurement.h"

//#include "DCtest.h"

//#include "ENC_packet.h"
//#include "ADC_packet.h"
//#include "IMU_Fill_packet.h"

//Define Global Variables
int waypoint = 0;     // For Debug

/***********DC Motor Variables***********/
float Power[4] = {20, 20, 20, 20};
float WheelVelocity[4] = {0, 0, 0, 0};
float VelocityError[4] = {0, 0, 0, 0};
float VelocitySetpoint[4] = {0, 0, 0, 0};
float IntegralError[4] = {0, 0, 0, 0};

/*********Stepper Motor Variables*********/
//uint32_t CCR[4] = {21000,21000,21000,21000};
//uint32_t CCRs[4] = {60000,60000,0,60000};
uint32_t CCRs[4] = {0,60000,0,0};
uint32_t CCRper[4] = {5000,15000,15000,15000};
float SteeringAngle[4] = {0, 0, 0, 0};
```

```
      float SteeringSetpoint[4]= {0, 0, 0, 0};
      uint32_t StepperAction[4] = {1,1,1,1};

      /*********IMU Variables*********/
      float       IMU[4] = {0,0,0,0};

      TM_RTC_Time_t Time;

      int main(void) {
            // Setup
            SystemInit();
            RCC_HSEConfig(RCC_HSE_ON);
            while(!RCC_WaitForHSEStartUp()){;}

            SimplePacket sp;
            SP_reset(&sp);

            TM_DISCO_LedInit();
            TM_DISCO_SetLed(LED_RED, HIGH);

            TM_USB_VCP_Init();                //initialize the Virtual
COM Port                                                    //KEK edit
            while(TM_USB_VCP_GetStatus() != TM_USB_VCP_CONNECTED) { ; }
// wait for successful connection   //KEK edit
            TM_DELAY_Init();

            TM_DISCO_SetLed(LED_RED,   LOW);
            TM_DISCO_SetLed(LED_GREEN, HIGH); // USB is good to go

            /**********************************************************
****************************/
                  //DC Motor Configurations
            //DC_GPIO_Initializations();
            //DC_TIM_Init();           //Initialize the DC motor TIM
ports

            /**********************************************************
****************************/
                  //IMU Initualizations
            //IMU_Initializations(); //Initialize the IMU pins
            //Zero_IMU_Rate_Bias();

            /**********************************************************
****************************/
                  //Encoder Configurations
            //GPIO__Encoder__and_TIM__Initializations();

            /**********************************************************
****************************/
                  //ADC Configurations
            //RCC_Configuration();
            //GPIO__Potentiometer__Initializations();
```

```
          //ADC_Configuration();
          //DMA_Configuration(motorValues);
          //ADC_DMARequestAfterLastTransferCmd(ADC1, ENABLE);
          //ADC_DMACmd(ADC1, ENABLE);
          //ADC_Cmd(ADC1, ENABLE);
          //ADC_SoftwareStartConv(ADC1);


/*****************************************************************
****************/
               //Stepper Configurations
          //Stepper_GPIO_Initialization();
          //Stepper_TIM_9_Init();
          //Stepper_TIM_10_Init();
          //Stepper_TIM_11_Init();
          //Stepper_TIM_12_Init();

          /*******************************************************
***************************/
               //RTC Initialization
          //TM_RTC_Init(TM_RTC_ClockSource_Internal);
          /*******************************************************
***************************/

          //float SteeringStart[4];
          //SteeringMotorAngles(SteeringStart, &waypoint, 0);
          //SteeringMotorAngles(SteeringAngle);
          //Delayms(3000);

          while(1) {
               //SteeringMotorAngles(SteeringAngle);
               //DC_Motor_Actuate(Power);

     //Get_Wheel_Velocity(WheelVelocity,VelocitySetpoint,VelocityError
);
               //DC_MOTOR_PID(Power,VelocityError,IntegralError);

               //Get_IMU_Data(IMU);

               char c;
               if(TM_USB_VCP_Getc(&c) == TM_USB_VCP_DATA_OK) { //
wait for data request

                    int i;
                    SP_reset(&sp); // clear the buffer
                    for(i = 0; i < 4; i++)
                         SP_addInt(&sp, 3 * i); // testing with
known values

                    //SteeringMotorAngles(SteeringAngle);
```

```
//Stepper_Motor_Action_Determination(SteeringAngle,SteeringSetpoi
nt,StepperAction);
                //Stepper_Motor_Actuate(CCRs,StepperAction);

                // send data if VCP port is ready
                if(TM_USB_VCP_GetStatus() ==
TM_USB_VCP_CONNECTED) {
                     TM_USB_VCP_Send(sp.buffer, sp.buf_size);

                     // indicate good USB connection
                     TM_DISCO_SetLed(LED_GREEN,  HIGH);
                     TM_DISCO_SetLed(LED_ORANGE, LOW);
                } else {
                     // indicate USB connection error
                     TM_DISCO_SetLed(LED_GREEN,  LOW);
                     TM_DISCO_SetLed(LED_ORANGE, HIGH);
                }
           }
      }

      // lol, return to what!?
      return 0;
 }
```

A.3.2. tm_stm32f4_delay

```
 /**
  * @author  Tilen Majerle
  * @email   tilen@majerle.eu
  * @website http://stm32f4-discovery.com
  * @link    http://stm32f4-discovery.com/2014/04/library-03-
stm32f429-discovery-system-clock-and-pretty-precise-delay-library/
  * @version v2.4
  * @ide     Keil uVision
  * @license GNU GPL v3
  * @brief   Pretty accurate delay functions with SysTick or any
other timer
  *
 @verbatim
   ---------------------------------------------------------------
--------
    Copyright (C) Tilen Majerle, 2015

    This program is free software: you can redistribute it and/or
modify
    it under the terms of the GNU General Public License as
published by
    the Free Software Foundation, either version 3 of the
License, or
    any later version.
```

```
         This program is distributed in the hope that it will be
useful,
         but WITHOUT ANY WARRANTY; without even the implied warranty
of
         MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
         GNU General Public License for more details.

         You should have received a copy of the GNU General Public
License
         along with this program.  If not, see
<http://www.gnu.org/licenses/>.
        ----------------------------------------------------------------
--------
     @endverbatim
      */
     #ifndef TM_DELAY_H
     #define TM_DELAY_H 240

     /* C++ detection */
     #ifdef __cplusplus
     extern "C" {
     #endif

     /**
      * @addtogroup TM_STM32F4xx_Libraries
      * @{
      */

     /**
      * @defgroup TM_DELAY
      * @brief    Pretty accurate delay functions with SysTick or any
other timer - http://stm32f4-discovery.com/2014/04/library-03-
stm32f429-discovery-system-clock-and-pretty-precise-delay-library/
      * @{
      *
     @verbatim
     If you are using GCC compiler, then your microseconds delay is
probably totally inaccurate.
     USE TIMER FOR DELAY, otherwise your delay will not be accurate.

     Another way is to use ARM compiler.
     @endverbatim
      *
      * As of version 2.0 you have now two possible ways to make a
delay.
      * The first (and default) is Systick timer. It makes interrupts
every 1ms.
      * If you want delay in "us" accuracy, there is simple pooling
(variable) mode.
      *
      *
```

```
      * The second (better) options is to use one of timers on F4xx
MCU.
      * Timer also makes an interrupts every 1ms (for count time)
instead of 1us as it was before.
      * For "us" delay, timer's counter is used to count ticks. It
makes a new tick each "us".
      * Not all MCUs have all possible timers, so this lib has been
designed that you select your own.
      *
      * \par Select custom TIM for delay functions
      *
      * By default, Systick timer is used for delay. If you want your
custom timer,
      * open defines.h file, add lines below and edit for your needs.
      *
     \code{.c}
     //Select custom timer for delay, here is TIM2 selected.
     //If you want custom TIMx, just replace number "2" for your TIM's
number.
     #define TM_DELAY_TIM                  TIM2
     #define TM_DELAY_TIM_IRQ              TIM2_IRQn
     #define TM_DELAY_TIM_IRQ_HANDLER TIM2_IRQHandler
     \endcode
      *
      *
      * With this setting (using custom timer) you have better
accuracy in "us" delay.
      * Also, you have to know, that if you want to use timer for
delay, you have to include additional files:
      *
      *    - CMSIS:
      *         - STM32F4xx TIM
      *         - MISC
      *    - TM:
      *         TM TIMER PROPERTIES
      *
      * Delay functions (Delay, Delayms) are now Inline functions.
      * This allows faster execution and more accurate delay.
      *
      * If you are working with Keil uVision and you are using Systick
for delay,
      * then set KEIL_IDE define in options for project:
      *    - Open "Options for target"
      *    - Tab "C/C++"
      *    - Under "Define" add "KEIL_IDE", without quotes
      *
      * \par Custom timers
      *
      * Custom timers are a way to make some tasks in a periodic
value.
      * As of version 2.4, delay library allows you to create custom
timer which count DOWN and when it reaches zero, callback is called.
```

```
       *
       * You can use variable settings for count, reload value and auto
reload feature.
       *
       * \par Changelog
       *
      @verbatim
       Version 2.4
        - May 26, 2015
        - Added support for custom timers which can be called
periodically

       Version 2.3
        - April 18, 2015
        - Fixed support for internal RC clock

       Version 2.2
        - January 12, 2015
        - Added support for custom function call each time 1ms
interrupt happen
        - Function is called TM_DELAY_1msHandler(void), with __weak
parameter
        - attributes.h file needed

       Version 2.1
        - GCC compiler fixes
        - Still prefer that you use TIM for delay if you are working
with ARM-GCC compiler

       Version 2.0
        - November 28, 2014
        - Delay library has been totally rewritten. Because Systick is
designed to be used
             in RTOS, it is not compatible to use it at the 2 places at
the same time.
             For that purpose, library has been rewritten.
        - Read full documentation above

       Version 1.0
        - First release
      @endverbatim
       *
       * \par Dependencies
       *
      @verbatim
       - STM32F4xx
       - STM32F4xx RCC:       Only if you want to use TIMx for delay
instead of Systick
       - STM32F4xx TIM:       Only if you want to use TIMx for delay
instead of Systick
       - MISC
       - defines.h
```

```
       - TM TIMER PROPERTIES: Only if you want to use TIMx for delay
instead of Systick
       - attribute.h
      @endverbatim
       */
      #include "stm32f4xx.h"
      #include "stm32f4xx_rcc.h"
      #include "defines.h"
      #include "attributes.h"
      /* If user selectable timer is selected for delay */
      #if defined(TM_DELAY_TIM)
      #include "misc.h"
      #include "stm32f4xx_tim.h"
      #include "tm_stm32f4_timer_properties.h"
      #endif
      #include "stdlib.h"

      /**
       * @defgroup TM_DELAY_Typedefs
       * @brief    Library Typedefs
       * @{
       */

      /**
       * @brief  Custom timer structure
       */
      typedef struct {
           uint32_t ARR;             /*!< Auto reload value */
           uint32_t AutoReload;      /*!< Set to 1 if timer should be
auto reloaded when it reaches zero */
           uint32_t CNT;             /*!< Counter value, counter
counts down */
           uint8_t Enabled;          /*!< Set to 1 when timer is
enabled */
           void (*Callback)(void *); /*!< Callback which will be
called when timer reaches zero */
           void* UserParameters;     /*!< Pointer to user parameters
used for callback function */
      } TM_DELAY_Timer_t;

      /**
       * @}
       */

      /**
       * @defgroup TM_DELAY_Macros
       * @brief    Library Macros
       * @{
       */

      /**
       * @brief  Number of allowed custom timers
```

```
 * @note   Should be changes in defines.h file if necessary
 */
#ifndef DELAY_MAX_CUSTOM_TIMERS
#define DELAY_MAX_CUSTOM_TIMERS   5
#endif

/* Memory allocation function */
#ifndef LIB_ALLOC_FUNC
#define LIB_ALLOC_FUNC    malloc
#endif

/* Memory free function */
#ifndef LIB_FREE_FUNC
#define LIB_FREE_FUNC     free
#endif

/**
 * @}
 */

/**
 * @defgroup TM_DELAY_Variables
 * @brief    Library Variables
 * @{
 */

/**
 * This variable can be used in main
 * It is automatically increased every time systick make an
interrupt
 */
extern __IO uint32_t TM_Time;
extern __IO uint32_t TM_Time2;
extern __IO uint32_t mult;

/**
 * @}
 */

/**
 * @defgroup TM_DELAY_Functions
 * @brief    Library Functions
 * @{
 */

/**
 * @param  Delays for specific amount of microseconds
 * @param  micros: Time in microseconds for delay
 * @retval None
 * @note   Declared as static inline
 */
static __INLINE void Delay(uint32_t micros) {
```

```
#if defined(TM_DELAY_TIM)
      volatile uint32_t timer = TM_DELAY_TIM->CNT;

      do {
            /* Count timer ticks */
            while ((TM_DELAY_TIM->CNT - timer) == 0);

            /* Increase timer */
            timer = TM_DELAY_TIM->CNT;

            /* Decrease microseconds */
      } while (--micros);
#else
      uint32_t amicros;

      /* Multiply micro seconds */
      amicros = (micros) * (mult);

      #ifdef __GNUC__
            if (SystemCoreClock == 180000000 || SystemCoreClock ==
100000000) {
                  amicros -= mult;
            }
      #endif

      /* If clock is 100MHz, then add additional multiplier */
      /* 100/3 = 33.3 = 33 and delay wouldn't be so accurate */
      #if defined(STM32F411xE)
      amicros += mult;
      #endif

      /* While loop */
      while (amicros--);
#endif /* TM_DELAY_TIM */
}

/**
 * @param  Delays for specific amount of milliseconds
 * @param  millis: Time in milliseconds for delay
 * @retval None
 * @note   Declared as static inline
 */
static __INLINE void Delayms(uint32_t millis) {
      volatile uint32_t timer = TM_Time;

      /* Called from thread */
      if (!__get_IPSR()) {
            /* Wait for timer to count milliseconds */
            while ((TM_Time - timer) < millis) {
#ifdef DELAY_SLEEP
                  /* Go sleep, wait systick interrupt */
                  __WFI();
```

```
#endif
            }
        } else {
            /* Called from interrupt */
            while (millis) {
                if (SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk) {
                    millis--;
                }
            }
        }
    }

    /**
     * @brief  Initializes timer settings for delay
     * @note   This function will initialize Systick or user timer,
according to settings
     * @param  None
     * @retval None
     */
    void TM_DELAY_Init(void);

    /**
     * @brief  Gets the TM_Time variable value
     * @param  None
     * @retval Current time in milliseconds
     */
    #define TM_DELAY_Time()                          (TM_Time)

    /**
     * @brief  Sets value for TM_Time variable
     * @param  time: Time in milliseconds
     * @retval None
     */
    #define TM_DELAY_SetTime(time)              (TM_Time = (time))

    /**
     * @brief  Re-enables delay timer It has to be configured before
with TM_DELAY_Init()
     * @note   This function enables delay timer. It can be systick
or user selectable timer.
     * @param  None
     * @retval None
     */
    void TM_DELAY_EnableDelayTimer(void);

    /**
     * @brief  Disables delay timer
     * @note   This function disables delay timer. It can be systick
or user selectable timer.
     * @param  None
     * @retval None
     */
```

```
void TM_DELAY_DisableDelayTimer(void);

/**
 * @brief  Gets the TM_Time2 variable value
 * @param  None
 * @retval Current time in milliseconds
 * @note   This is not meant for public use
 */
#define TM_DELAY_Time2()                    (TM_Time2)

/**
 * @brief  Sets value for TM_Time variable
 * @param  time: Time in milliseconds
 * @retval None
 * @note   This is not meant for public use
 */
#define TM_DELAY_SetTime2(time)             (TM_Time2 = (time))

/**
 * @brief  Creates a new custom timer which has 1ms resolution
 * @note   It uses @ref malloc for memory allocation for timer
structure
 * @param  ReloadValue: Number of milliseconds when timer reaches
zero and callback function is called
 * @param  AutoReload: If set to 1, timer will start again when
it reaches zero and callback is called
 * @param  StartTimer: If set to 1, timer will start immediately
 * @param  *TM_DELAY_CustomTimerCallback: Pointer to callback
function which will be called when timer reaches zero
 * @param  *UserParameters: Pointer to void pointer to user
parameters used as first parameter in callback function
 * @retval Pointer to allocated timer structure
 */
TM_DELAY_Timer_t* TM_DELAY_TimerCreate(uint32_t ReloadValue,
uint8_t AutoReload, uint8_t StartTimer, void
(*TM_DELAY_CustomTimerCallback)(void *), void* UserParameters);

/**
 * @brief  Deletes already allocated timer
 * @param  *Timer: Pointer to @ref TM_DELAY_Timer_t structure
 * @retval None
 */
void TM_DELAY_TimerDelete(TM_DELAY_Timer_t* Timer);

/**
 * @brief  Stops custom timer from counting
 * @param  *Timer: Pointer to @ref TM_DELAY_Timer_t structure
 * @retval Pointer to @ref TM_DELAY_Timer_t structure
 */
TM_DELAY_Timer_t* TM_DELAY_TimerStop(TM_DELAY_Timer_t* Timer);

/**
```

```
 * @brief  Starts custom timer counting
 * @param  *Timer: Pointer to @ref TM_DELAY_Timer_t structure
 * @retval Pointer to @ref TM_DELAY_Timer_t structure
 */
TM_DELAY_Timer_t* TM_DELAY_TimerStart(TM_DELAY_Timer_t* Timer);

/**
 * @brief  Resets custom timer counter value
 * @param  *Timer: Pointer to @ref TM_DELAY_Timer_t structure
 * @retval Pointer to @ref TM_DELAY_Timer_t structure
 */
TM_DELAY_Timer_t* TM_DELAY_TimerReset(TM_DELAY_Timer_t* Timer);

/**
 * @brief  Sets auto reload feature for timer
 * @note   Auto reload features is used for timer which starts
again when zero is reached if auto reload active
 * @param  *Timer: Pointer to @ref TM_DELAY_Timer_t structure
 * uint8_t AutoReload: Set to 1 if you want to enable AutoReload
or 0 to disable
 * @retval Pointer to @ref TM_DELAY_Timer_t structure
 */
TM_DELAY_Timer_t* TM_DELAY_TimerAutoReload(TM_DELAY_Timer_t*
Timer, uint8_t AutoReload);

/**
 * @brief  Sets auto reload value for timer
 * @param  *Timer: Pointer to @ref TM_DELAY_Timer_t structure
 * @param  AutoReloadValue: Value for timer to be set when zero
is reached and callback is called
 * @note   AutoReload feature must be enabled for timer in order
to get this to work properly
 * @retval Pointer to @ref TM_DELAY_Timer_t structure
 */
TM_DELAY_Timer_t* TM_DELAY_TimerAutoReloadValue(TM_DELAY_Timer_t*
Timer, uint32_t AutoReloadValue);

/**
 * @brief  User function, called each 1ms when interrupt from
timer happen
 * @note   Here user should put things which has to be called
periodically
 * @param  None
 * @retval None
 * @note   With __weak parameter to prevent link errors if not
defined by user
 */
__weak void TM_DELAY_1msHandler(void);


/**
 * @}
```

```
 */

/**
 * @}
 */

/**
 * @}
 */

/* C++ detection */
#ifdef __cplusplus
}
#endif

#endif
```

A.3.3. tm_stm32f4_gpio

```
/**
 * |----------------------------------------------------------------------
---------
 * | Copyright (C) Tilen Majerle, 2015
 * |
 * | This program is free software: you can redistribute it
and/or modify
 * | it under the terms of the GNU General Public License as
published by
 * | the Free Software Foundation, either version 3 of the
License, or
 * | any later version.
 * |
 * | This program is distributed in the hope that it will be
useful,
 * | but WITHOUT ANY WARRANTY; without even the implied warranty
of
 * | MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See
the
 * | GNU General Public License for more details.
 * |
 * | You should have received a copy of the GNU General Public
License
 * | along with this program.  If not, see
<http://www.gnu.org/licenses/>.
 * |----------------------------------------------------------------------
---------
 */
#include "tm_stm32f4_gpio.h"

/* Private function */
static uint16_t GPIO_UsedPins[11] = {0,0,0,0,0,0,0,0,0,0,0};
```

```
     /* Private functions */
     void TM_GPIO_INT_EnableClock(GPIO_TypeDef* GPIOx);
     void TM_GPIO_INT_DisableClock(GPIO_TypeDef* GPIOx);
     void TM_GPIO_INT_Init(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin,
TM_GPIO_Mode_t GPIO_Mode, TM_GPIO_OType_t GPIO_OType, TM_GPIO_PuPd_t
GPIO_PuPd, TM_GPIO_Speed_t GPIO_Speed);

     void TM_GPIO_Init(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin,
TM_GPIO_Mode_t GPIO_Mode, TM_GPIO_OType_t GPIO_OType, TM_GPIO_PuPd_t
GPIO_PuPd, TM_GPIO_Speed_t GPIO_Speed) {
          /* Check input */
          if (GPIO_Pin == 0x00) {
               return;
          }

          /* Enable clock for GPIO */
          TM_GPIO_INT_EnableClock(GPIOx);

          /* Do initialization */
          TM_GPIO_INT_Init(GPIOx, GPIO_Pin, GPIO_Mode, GPIO_OType,
GPIO_PuPd, GPIO_Speed);
     }

     void TM_GPIO_InitAlternate(GPIO_TypeDef* GPIOx, uint16_t
GPIO_Pin, TM_GPIO_OType_t GPIO_OType, TM_GPIO_PuPd_t GPIO_PuPd,
TM_GPIO_Speed_t GPIO_Speed, uint8_t Alternate) {
          uint32_t pinpos;

          /* Check input */
          if (GPIO_Pin == 0x00) {
               return;
          }

          /* Enable GPIOx clock */
          TM_GPIO_INT_EnableClock(GPIOx);

          /* Set alternate functions for all pins */
          for (pinpos = 0; pinpos < 0x10; pinpos++) {
               /* Check pin */
               if ((GPIO_Pin & (1 << pinpos)) == 0) {
                    continue;
               }

               /* Set alternate function */
               GPIOx->AFR[pinpos >> 0x03] = (GPIOx->AFR[pinpos >>
0x03] & ~(0x0F << (4 * (pinpos & 0x07)))) | (Alternate << (4 * (pinpos
& 0x07)));
          }

          /* Do initialization */
```

```
		TM_GPIO_INT_Init(GPIOx, GPIO_Pin, TM_GPIO_Mode_AF,
GPIO_OType, GPIO_PuPd, GPIO_Speed);
	}

	void TM_GPIO_DeInit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin) {
		uint8_t i;
		uint8_t ptr = TM_GPIO_GetPortSource(GPIOx);

		/* Go through all pins */
		for (i = 0x00; i < 0x10; i++) {
			/* Pin is set */
			if (GPIO_Pin & (1 << i)) {
				/* Set 11 bits combination for analog mode */
				GPIOx->MODER |= (0x03 << (2 * i));

				/* Pin is not used */
				GPIO_UsedPins[ptr] &= ~(1 << i);
			}
		}
	}

	void TM_GPIO_SetPinAsInput(GPIO_TypeDef* GPIOx, uint16_t
GPIO_Pin) {
		uint8_t i;
		/* Go through all pins */
		for (i = 0x00; i < 0x10; i++) {
			/* Pin is set */
			if (GPIO_Pin & (1 << i)) {
				/* Set 00 bits combination for input */
				GPIOx->MODER &= ~(0x03 << (2 * i));
			}
		}
	}

	void TM_GPIO_SetPinAsOutput(GPIO_TypeDef* GPIOx, uint16_t
GPIO_Pin) {
		uint8_t i;
		/* Go through all pins */
		for (i = 0x00; i < 0x10; i++) {
			/* Pin is set */
			if (GPIO_Pin & (1 << i)) {
				/* Set 01 bits combination for output */
				GPIOx->MODER = (GPIOx->MODER & ~(0x03 << (2 *
i))) | (0x01 << (2 * i));
			}
		}
	}

	void TM_GPIO_SetPinAsAnalog(GPIO_TypeDef* GPIOx, uint16_t
GPIO_Pin) {
		uint8_t i;
		/* Go through all pins */
```

```
		for (i = 0x00; i < 0x10; i++) {
			/* Pin is set */
			if (GPIO_Pin & (1 << i)) {
				/* Set 11 bits combination for analog mode */
				GPIOx->MODER |= (0x03 << (2 * i));
			}
		}
	}

	void TM_GPIO_SetPinAsAlternate(GPIO_TypeDef* GPIOx, uint16_t
GPIO_Pin) {
		uint8_t i;

		/* Set alternate functions for all pins */
		for (i = 0; i < 0x10; i++) {
			/* Check pin */
			if ((GPIO_Pin & (1 << i)) == 0) {
				continue;
			}

			/* Set alternate mode */
			GPIOx->MODER = (GPIOx->MODER & ~(0x03 << (2 * i))) |
(0x02 << (2 * i));
		}
	}

	void TM_GPIO_SetPullResistor(GPIO_TypeDef* GPIOx, uint16_t
GPIO_Pin, TM_GPIO_PuPd_t GPIO_PuPd) {
		uint8_t pinpos;

		/* Go through all pins */
		for (pinpos = 0; pinpos < 0x10; pinpos++) {
			/* Check if pin available */
			if ((GPIO_Pin & (1 << pinpos)) == 0) {
				continue;
			}

			/* Set GPIO PUPD register */
			GPIOx->PUPDR = (GPIOx->PUPDR & ~(0x03 << (2 *
pinpos))) | ((uint32_t)(GPIO_PuPd << (2 * pinpos)));
		}
	}

	void TM_GPIO_Lock(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin) {
		uint32_t d;

		/* Set GPIO pin with 16th bit set to 1 */
		d = 0x00010000 | GPIO_Pin;

		/* Write to LCKR register */
		GPIOx->LCKR = d;
		GPIOx->LCKR = GPIO_Pin;
```

```
        GPIOx->LCKR = d;

        /* Read twice */
        (void)GPIOx->LCKR;
        (void)GPIOx->LCKR;
    }

    uint16_t TM_GPIO_GetPinSource(uint16_t GPIO_Pin) {
        uint16_t pinsource = 0;

        /* Get pinsource */
        while (GPIO_Pin > 1) {
            /* Increase pinsource */
            pinsource++;
            /* Shift right */
            GPIO_Pin >>= 1;
        }

        /* Return source */
        return pinsource;
    }

    uint16_t TM_GPIO_GetPortSource(GPIO_TypeDef* GPIOx) {
        /* Get port source number */
        /* Offset from GPIOA                       Difference
between 2 GPIO addresses */
        return ((uint32_t)GPIOx - (GPIOA_BASE)) / ((GPIOB_BASE) -
(GPIOA_BASE));
    }

    /* Private functions */
    void TM_GPIO_INT_EnableClock(GPIO_TypeDef* GPIOx) {
        /* Set bit according to the 1 << portsourcenumber */
        RCC->AHB1ENR |= (1 << TM_GPIO_GetPortSource(GPIOx));
    }

    void TM_GPIO_INT_DisableClock(GPIO_TypeDef* GPIOx) {
        /* Clear bit according to the 1 << portsourcenumber */
        RCC->AHB1ENR &= ~(1 << TM_GPIO_GetPortSource(GPIOx));
    }

    void TM_GPIO_INT_Init(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin,
TM_GPIO_Mode_t GPIO_Mode, TM_GPIO_OType_t GPIO_OType, TM_GPIO_PuPd_t
GPIO_PuPd, TM_GPIO_Speed_t GPIO_Speed) {
        uint8_t pinpos;
        uint8_t ptr = TM_GPIO_GetPortSource(GPIOx);

        /* Go through all pins */
        for (pinpos = 0; pinpos < 0x10; pinpos++) {
            /* Check if pin available */
            if ((GPIO_Pin & (1 << pinpos)) == 0) {
                continue;
```

```
                }

                /* Pin is used */
                GPIO_UsedPins[ptr] |= 1 << pinpos;

                /* Set GPIO PUPD register */
                GPIOx->PUPDR = (GPIOx->PUPDR & ~(0x03 << (2 *
pinpos))) | ((uint32_t)(GPIO_PuPd << (2 * pinpos)));

                /* Set GPIO MODE register */
                GPIOx->MODER = (GPIOx->MODER & ~((uint32_t)(0x03 << (2
* pinpos)))) | ((uint32_t)(GPIO_Mode << (2 * pinpos)));

                /* Set only if output or alternate functions */
                if (GPIO_Mode == TM_GPIO_Mode_OUT || GPIO_Mode ==
TM_GPIO_Mode_AF) {
                      /* Set GPIO OTYPE register */
                      GPIOx->OTYPER = (GPIOx->OTYPER & ~(uint16_t)(0x01
<< pinpos)) | ((uint16_t)(GPIO_OType << pinpos));

                      /* Set GPIO OSPEED register */
                      GPIOx->OSPEEDR = (GPIOx->OSPEEDR &
~((uint32_t)(0x03 << (2 * pinpos)))) | ((uint32_t)(GPIO_Speed << (2 *
pinpos)));
                }
          }
    }

    uint16_t TM_GPIO_GetUsedPins(GPIO_TypeDef* GPIOx) {
          /* Return used */
          return GPIO_UsedPins[TM_GPIO_GetPortSource(GPIOx)];
    }

    uint16_t TM_GPIO_GetFreePins(GPIO_TypeDef* GPIOx) {
          /* Return free pins */
          return ~GPIO_UsedPins[TM_GPIO_GetPortSource(GPIOx)];
    }
```

A.3.4. tm_stm32f4_spi

```
    /**
     * |----------------------------------------------------------------------
---------
     * | Copyright (C) Tilen Majerle, 2014
     * |
     * | This program is free software: you can redistribute it
and/or modify
     * | it under the terms of the GNU General Public License as
published by
     * | the Free Software Foundation, either version 3 of the
License, or
```

```
       * | any later version.
       * |
       * | This program is distributed in the hope that it will be
useful,
       * | but WITHOUT ANY WARRANTY; without even the implied warranty
of
       * | MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See
the
       * | GNU General Public License for more details.
       * |
       * | You should have received a copy of the GNU General Public
License
       * | along with this program.  If not, see
<http://www.gnu.org/licenses/>.
       * |----------------------------------------------------------------
---------
       */
      #include "tm_stm32f4_spi.h"

      /* Private functions */
      static void TM_SPIx_Init(SPI_TypeDef* SPIx, TM_SPI_PinsPack_t
pinspack, TM_SPI_Mode_t SPI_Mode, uint16_t SPI_BaudRatePrescaler,
uint16_t SPI_MasterSlave, uint16_t SPI_FirstBit);
      void TM_SPI1_INT_InitPins(TM_SPI_PinsPack_t pinspack);
      void TM_SPI2_INT_InitPins(TM_SPI_PinsPack_t pinspack);
      void TM_SPI3_INT_InitPins(TM_SPI_PinsPack_t pinspack);
      void TM_SPI4_INT_InitPins(TM_SPI_PinsPack_t pinspack);
      void TM_SPI5_INT_InitPins(TM_SPI_PinsPack_t pinspack);
      void TM_SPI6_INT_InitPins(TM_SPI_PinsPack_t pinspack);

      void TM_SPI_Init(SPI_TypeDef* SPIx, TM_SPI_PinsPack_t pinspack) {
            /* Init with default settings */
      #ifdef USE_SPI1
            if (SPIx == SPI1) {
                  TM_SPIx_Init(SPI1, pinspack, TM_SPI1_MODE,
TM_SPI1_PRESCALER, TM_SPI1_MASTERSLAVE, TM_SPI1_FIRSTBIT);
            }
      #endif
      #ifdef USE_SPI2
            if (SPIx == SPI2) {
                  TM_SPIx_Init(SPI2, pinspack, TM_SPI2_MODE,
TM_SPI2_PRESCALER, TM_SPI2_MASTERSLAVE, TM_SPI2_FIRSTBIT);
            }
      #endif
      #ifdef USE_SPI3
            if (SPIx == SPI3) {
                  TM_SPIx_Init(SPI3, pinspack, TM_SPI3_MODE,
TM_SPI3_PRESCALER, TM_SPI3_MASTERSLAVE, TM_SPI3_FIRSTBIT);
            }
      #endif
      #ifdef USE_SPI4
            if (SPIx == SPI4) {
```

```
                TM_SPIx_Init(SPI4, pinspack, TM_SPI4_MODE,
TM_SPI4_PRESCALER, TM_SPI4_MASTERSLAVE, TM_SPI4_FIRSTBIT);
          }
     #endif
     #ifdef USE_SPI5
          if (SPIx == SPI5) {
                TM_SPIx_Init(SPI5, pinspack, TM_SPI5_MODE,
TM_SPI5_PRESCALER, TM_SPI5_MASTERSLAVE, TM_SPI5_FIRSTBIT);
          }
     #endif
     #ifdef USE_SPI6
          if (SPIx == SPI6) {
                TM_SPIx_Init(SPI6, pinspack, TM_SPI6_MODE,
TM_SPI6_PRESCALER, TM_SPI6_MASTERSLAVE, TM_SPI6_FIRSTBIT);
          }
     #endif
     }

     void TM_SPI_InitWithMode(SPI_TypeDef* SPIx, TM_SPI_PinsPack_t
pinspack, TM_SPI_Mode_t SPI_Mode) {
          /* Init with custom mode, 0, 1, 2, 3 */
     #ifdef USE_SPI1
          if (SPIx == SPI1) {
                TM_SPIx_Init(SPI1, pinspack, SPI_Mode,
TM_SPI1_PRESCALER, TM_SPI1_MASTERSLAVE, TM_SPI1_FIRSTBIT);
          }
     #endif
     #ifdef USE_SPI2
          if (SPIx == SPI2) {
                TM_SPIx_Init(SPI2, pinspack, SPI_Mode,
TM_SPI2_PRESCALER, TM_SPI2_MASTERSLAVE, TM_SPI2_FIRSTBIT);
          }
     #endif
     #ifdef USE_SPI3
          if (SPIx == SPI3) {
                TM_SPIx_Init(SPI3, pinspack, SPI_Mode,
TM_SPI3_PRESCALER, TM_SPI3_MASTERSLAVE, TM_SPI3_FIRSTBIT);
          }
     #endif
     #ifdef USE_SPI4
          if (SPIx == SPI4) {
                TM_SPIx_Init(SPI4, pinspack, SPI_Mode,
TM_SPI4_PRESCALER, TM_SPI4_MASTERSLAVE, TM_SPI4_FIRSTBIT);
          }
     #endif
     #ifdef USE_SPI5
          if (SPIx == SPI5) {
                TM_SPIx_Init(SPI5, pinspack, SPI_Mode,
TM_SPI5_PRESCALER, TM_SPI5_MASTERSLAVE, TM_SPI5_FIRSTBIT);
          }
     #endif
     #ifdef USE_SPI6
```

```
		if (SPIx == SPI6) {
			TM_SPIx_Init(SPI6, pinspack, SPI_Mode,
TM_SPI6_PRESCALER, TM_SPI6_MASTERSLAVE, TM_SPI6_FIRSTBIT);
		}
	#endif
	}

	void TM_SPI_InitFull(
		SPI_TypeDef* SPIx,                \
		TM_SPI_PinsPack_t pinspack,       \
		uint16_t SPI_BaudRatePrescaler,   \
		TM_SPI_Mode_t SPI_Mode_t,         \
		uint16_t SPI_Mode,                \
		uint16_t SPI_FirstBit             \
	) {
		/* Init FULL SPI settings by user */
	#ifdef USE_SPI1
		if (SPIx == SPI1) {
			TM_SPIx_Init(SPI1, pinspack, SPI_Mode_t,
SPI_BaudRatePrescaler, SPI_Mode, SPI_FirstBit);
		}
	#endif
	#ifdef USE_SPI2
		if (SPIx == SPI2) {
			TM_SPIx_Init(SPI2, pinspack, SPI_Mode_t,
SPI_BaudRatePrescaler, SPI_Mode, SPI_FirstBit);
		}
	#endif
	#ifdef USE_SPI3
		if (SPIx == SPI3) {
			TM_SPIx_Init(SPI3, pinspack, SPI_Mode_t,
SPI_BaudRatePrescaler, SPI_Mode, SPI_FirstBit);
		}
	#endif
	#ifdef USE_SPI4
		if (SPIx == SPI4) {
			TM_SPIx_Init(SPI4, pinspack, SPI_Mode_t,
SPI_BaudRatePrescaler, SPI_Mode, SPI_FirstBit);
		}
	#endif
	#ifdef USE_SPI5
		if (SPIx == SPI5) {
			TM_SPIx_Init(SPI5, pinspack, SPI_Mode_t,
SPI_BaudRatePrescaler, SPI_Mode, SPI_FirstBit);
		}
	#endif
	#ifdef USE_SPI6
		if (SPIx == SPI6) {
			TM_SPIx_Init(SPI6, pinspack, SPI_Mode_t,
SPI_BaudRatePrescaler, SPI_Mode, SPI_FirstBit);
		}
	#endif
```

```
	}

	uint16_t TM_SPI_GetPrescalerFromMaxFrequency(SPI_TypeDef* SPIx,
uint32_t MAX_SPI_Frequency) {
		RCC_ClocksTypeDef RCC_Clocks;
		uint32_t APB_Frequency;
		uint8_t i;

		/* Prevent false input */
		if (MAX_SPI_Frequency == 0) {
			return SPI_BaudRatePrescaler_256;
		}

		/* Get clock values from RCC */
		RCC_GetClocksFreq(&RCC_Clocks);

		/* Calculate max SPI clock */
		if (0
	#ifdef USE_SPI1
			|| SPIx == SPI1
	#endif
	#ifdef USE_SPI4
			|| SPIx == SPI4
	#endif
	#ifdef USE_SPI5
			|| SPIx == SPI5
	#endif
	#ifdef USE_SPI6
			|| SPIx == SPI6
	#endif
		) {
			APB_Frequency = RCC_Clocks.PCLK2_Frequency;
		} else {
			APB_Frequency = RCC_Clocks.PCLK1_Frequency;
		}

		/* Calculate prescaler value */
		/* Bits 5:3 in CR1 SPI registers are prescalers */
		/* 000 = 2, 001 = 4, 002 = 8, ..., 111 = 256 */
		for (i = 0; i < 8; i++) {
			if (APB_Frequency / (1 << (i + 1)) <=
MAX_SPI_Frequency) {
				/* Bits for BP are 5:3 in CR1 register */
				return (i << 3);
			}
		}

		/* Use max prescaler possible */
		return SPI_BaudRatePrescaler_256;
	}
```

```
      TM_SPI_DataSize_t TM_SPI_SetDataSize(SPI_TypeDef* SPIx,
TM_SPI_DataSize_t DataSize) {
            TM_SPI_DataSize_t status = (SPIx->CR1 & SPI_CR1_DFF) ?
TM_SPI_DataSize_16b : TM_SPI_DataSize_8b;

            /* Disable SPI first */
            SPIx->CR1 &= ~SPI_CR1_SPE;

            /* Set proper value */
            if (DataSize == TM_SPI_DataSize_16b) {
                  /* Set bit for frame */
                  SPIx->CR1 |= SPI_CR1_DFF;
            } else {
                  /* Clear bit for frame */
                  SPIx->CR1 &= ~SPI_CR1_DFF;
            }

            /* Enable SPI back */
            SPIx->CR1 |= SPI_CR1_SPE;

            /* Return status */
            return status;
      }

      void TM_SPI_SendMulti(SPI_TypeDef* SPIx, uint8_t* dataOut,
uint8_t* dataIn, uint32_t count) {
            uint32_t i;

            /* Check if SPI is enabled */
            SPI_CHECK_ENABLED(SPIx);

            /* Wait for previous transmissions to complete if DMA TX
enabled for SPI */
            SPI_WAIT(SPIx);

            for (i = 0; i < count; i++) {
                  /* Fill output buffer with data */
                  SPIx->DR = dataOut[i];

                  /* Wait for SPI to end everything */
                  SPI_WAIT(SPIx);

                  /* Read data register */
                  dataIn[i] = SPIx->DR;
            }
      }

      void TM_SPI_WriteMulti(SPI_TypeDef* SPIx, uint8_t* dataOut,
uint32_t count) {
            uint32_t i;

            /* Check if SPI is enabled */
```

```
            SPI_CHECK_ENABLED(SPIx);

            /* Wait for previous transmissions to complete if DMA TX
enabled for SPI */
            SPI_WAIT(SPIx);

            for (i = 0; i < count; i++) {
                  /* Fill output buffer with data */
                  SPIx->DR = dataOut[i];

                  /* Wait for SPI to end everything */
                  SPI_WAIT(SPIx);

                  /* Read data register */
                  (void)SPIx->DR;
            }
      }

      void TM_SPI_ReadMulti(SPI_TypeDef* SPIx, uint8_t* dataIn, uint8_t
dummy, uint32_t count) {
            uint32_t i;

            /* Check if SPI is enabled */
            SPI_CHECK_ENABLED(SPIx);

            /* Wait for previous transmissions to complete if DMA TX
enabled for SPI */
            SPI_WAIT(SPIx);

            for (i = 0; i < count; i++) {
                  /* Fill output buffer with data */
                  SPIx->DR = dummy;

                  /* Wait for SPI to end everything */
                  SPI_WAIT(SPIx);

                  /* Save data to buffer */
                  dataIn[i] = SPIx->DR;
            }
      }

      void TM_SPI_SendMulti16(SPI_TypeDef* SPIx, uint16_t* dataOut,
uint16_t* dataIn, uint32_t count) {
            uint32_t i;

            /* Check if SPI is enabled */
            SPI_CHECK_ENABLED(SPIx);

            /* Wait for previous transmissions to complete if DMA TX
enabled for SPI */
            SPI_WAIT(SPIx);
```

```
        for (i = 0; i < count; i++) {
            /* Fill output buffer with data */
            SPIx->DR = dataOut[i];

            /* Wait for SPI to end everything */
            SPI_WAIT(SPIx);

            /* Read data register */
            dataIn[i] = SPIx->DR;
        }
    }

    void TM_SPI_WriteMulti16(SPI_TypeDef* SPIx, uint16_t* dataOut,
uint32_t count) {
        uint32_t i;

        /* Check if SPI is enabled */
        SPI_CHECK_ENABLED(SPIx);

        /* Wait for previous transmissions to complete if DMA TX
enabled for SPI */
        SPI_WAIT(SPIx);

        for (i = 0; i < count; i++) {
            /* Fill output buffer with data */
            SPIx->DR = dataOut[i];

            /* Wait for SPI to end everything */
            SPI_WAIT(SPIx);

            /* Read data register */
            (void)SPIx->DR;
        }
    }

    void TM_SPI_ReadMulti16(SPI_TypeDef* SPIx, uint16_t* dataIn,
uint16_t dummy, uint32_t count) {
        uint32_t i;

        /* Check if SPI is enabled */
        SPI_CHECK_ENABLED(SPIx);

        /* Wait for previous transmissions to complete if DMA TX
enabled for SPI */
        SPI_WAIT(SPIx);

        for (i = 0; i < count; i++) {
            /* Fill output buffer with data */
            SPIx->DR = dummy;

            /* Wait for SPI to end everything */
            SPI_WAIT(SPIx);
```

```
            /* Save data to buffer */
            dataIn[i] = SPIx->DR;
        }
    }

    __weak void TM_SPI_InitCustomPinsCallback(SPI_TypeDef* SPIx,
uint16_t AlternateFunction) {
        /* Custom user function. */
        /* In case user needs functionality for custom pins, this
function should be declared outside this library */
    }

    /* Private functions */
    static void TM_SPIx_Init(SPI_TypeDef* SPIx, TM_SPI_PinsPack_t
pinspack, TM_SPI_Mode_t SPI_Mode, uint16_t SPI_BaudRatePrescaler,
uint16_t SPI_MasterSlave, uint16_t SPI_FirstBit) {
        SPI_InitTypeDef SPI_InitStruct;

        /* Set default settings */
        SPI_StructInit(&SPI_InitStruct);
    #ifdef USE_SPI1
        if (SPIx == SPI1) {
            /* Enable SPI clock */
            RCC->APB2ENR |= RCC_APB2ENR_SPI1EN;

            /* Init pins */
            TM_SPI1_INT_InitPins(pinspack);

            /* Set options */
            SPI_InitStruct.SPI_DataSize = TM_SPI1_DATASIZE;
        }
    #endif
    #ifdef USE_SPI2
        if (SPIx == SPI2) {
            /* Enable SPI clock */
            RCC->APB1ENR |= RCC_APB1ENR_SPI2EN;

            /* Init pins */
            TM_SPI2_INT_InitPins(pinspack);

            /* Set options */
            SPI_InitStruct.SPI_DataSize = TM_SPI2_DATASIZE;
        }
    #endif
    #ifdef USE_SPI3
        if (SPIx == SPI3) {
            /* Enable SPI clock */
            RCC->APB1ENR |= RCC_APB1ENR_SPI3EN;

            /* Init pins */
            TM_SPI3_INT_InitPins(pinspack);
```

```
            /* Set options */
            SPI_InitStruct.SPI_DataSize = TM_SPI3_DATASIZE;

        }
    #endif
    #ifdef USE_SPI4
        if (SPIx == SPI4) {
            /* Enable SPI clock */
            RCC->APB2ENR |= RCC_APB2ENR_SPI4EN;

            /* Init pins */
            TM_SPI4_INT_InitPins(pinspack);

            /* Set options */
            SPI_InitStruct.SPI_DataSize = TM_SPI4_DATASIZE;
        }
    #endif
    #ifdef USE_SPI5
        if (SPIx == SPI5) {
            /* Enable SPI clock */
            RCC->APB2ENR |= RCC_APB2ENR_SPI5EN;

            /* Init pins */
            TM_SPI5_INT_InitPins(pinspack);

            /* Set options */
            SPI_InitStruct.SPI_DataSize = TM_SPI5_DATASIZE;
        }
    #endif
    #ifdef USE_SPI6
        if (SPIx == SPI6) {
            /* Enable SPI clock */
            RCC->APB2ENR |= RCC_APB2ENR_SPI6EN;

            /* Init pins */
            TM_SPI6_INT_InitPins(pinspack);

            /* Set options */
            SPI_InitStruct.SPI_DataSize = TM_SPI6_DATASIZE;
        }
    #endif

        /* Fill SPI settings */
        SPI_InitStruct.SPI_BaudRatePrescaler =
SPI_BaudRatePrescaler;
        SPI_InitStruct.SPI_Direction =
SPI_Direction_2Lines_FullDuplex;
        SPI_InitStruct.SPI_FirstBit = SPI_FirstBit;
        SPI_InitStruct.SPI_Mode = SPI_MasterSlave;
        SPI_InitStruct.SPI_NSS = SPI_NSS_Soft;
        //SPI_InitStruct.SPI_DataSize = SPI_DataSize_16b;
```

```
		/* SPI mode */
		if (SPI_Mode == TM_SPI_Mode_0) {
			SPI_InitStruct.SPI_CPOL = SPI_CPOL_Low;
			SPI_InitStruct.SPI_CPHA = SPI_CPHA_1Edge;
		} else if (SPI_Mode == TM_SPI_Mode_1) {
			SPI_InitStruct.SPI_CPOL = SPI_CPOL_Low;
			SPI_InitStruct.SPI_CPHA = SPI_CPHA_2Edge;
		} else if (SPI_Mode == TM_SPI_Mode_2) {
			SPI_InitStruct.SPI_CPOL = SPI_CPOL_High;
			SPI_InitStruct.SPI_CPHA = SPI_CPHA_1Edge;
		} else if (SPI_Mode == TM_SPI_Mode_3) {
			SPI_InitStruct.SPI_CPOL = SPI_CPOL_High;
			SPI_InitStruct.SPI_CPHA = SPI_CPHA_2Edge;
		}

		/* Disable first */
		SPIx->CR1 &= ~SPI_CR1_SPE;

		/* Init SPI */
		SPI_Init(SPIx, &SPI_InitStruct);

		/* Enable SPI */
		SPIx->CR1 |= SPI_CR1_SPE;
	}

	/* Private functions */
	#ifdef USE_SPI1
	void TM_SPI1_INT_InitPins(TM_SPI_PinsPack_t pinspack) {
		/* Init SPI pins */
	#if defined(GPIOA)
		if (pinspack == TM_SPI_PinsPack_1) {
			TM_GPIO_InitAlternate(GPIOA, GPIO_PIN_5 | GPIO_PIN_6 |
GPIO_PIN_7, TM_GPIO_OType_PP, TM_GPIO_PuPd_NOPULL, TM_GPIO_Speed_High,
GPIO_AF_SPI1);
		}
	#endif
	#if defined(GPIOB)
		if (pinspack == TM_SPI_PinsPack_2) {
			TM_GPIO_InitAlternate(GPIOB, GPIO_PIN_3 | GPIO_PIN_4 |
GPIO_PIN_5, TM_GPIO_OType_PP, TM_GPIO_PuPd_NOPULL, TM_GPIO_Speed_High,
GPIO_AF_SPI1);
		}
	#endif
		if (pinspack == TM_SPI_PinsPack_Custom) {
			/* Call user function */
			TM_SPI_InitCustomPinsCallback(SPI1, GPIO_AF_SPI1);
		}
	}
	#endif

	#ifdef USE_SPI2
```

```
	void TM_SPI2_INT_InitPins(TM_SPI_PinsPack_t pinspack) {
		/* Init SPI pins */
	#if defined(GPIOB) && defined(GPIOC)
		if (pinspack == TM_SPI_PinsPack_1) {
			TM_GPIO_InitAlternate(GPIOB, GPIO_PIN_10,
TM_GPIO_OType_PP, TM_GPIO_PuPd_NOPULL, TM_GPIO_Speed_High,
GPIO_AF_SPI2);
			TM_GPIO_InitAlternate(GPIOC, GPIO_PIN_2 | GPIO_PIN_3,
TM_GPIO_OType_PP, TM_GPIO_PuPd_NOPULL, TM_GPIO_Speed_High,
GPIO_AF_SPI2);
		}
	#endif
	#if defined(GPIOB)
		if (pinspack == TM_SPI_PinsPack_2) {
			TM_GPIO_InitAlternate(GPIOB, GPIO_PIN_13 | GPIO_PIN_14
| GPIO_PIN_15, TM_GPIO_OType_PP, TM_GPIO_PuPd_NOPULL,
TM_GPIO_Speed_High, GPIO_AF_SPI2);
		}
	#endif
	#if defined(GPIOI)
		if (pinspack == TM_SPI_PinsPack_3) {
			TM_GPIO_InitAlternate(GPIOI, GPIO_PIN_0 | GPIO_PIN_2 |
GPIO_PIN_3, TM_GPIO_OType_PP, TM_GPIO_PuPd_NOPULL, TM_GPIO_Speed_High,
GPIO_AF_SPI2);
		}
	#endif
		if (pinspack == TM_SPI_PinsPack_Custom) {
			/* Call user function */
			TM_SPI_InitCustomPinsCallback(SPI2, GPIO_AF_SPI2);
		}
	}
	#endif

	#ifdef USE_SPI3
	void TM_SPI3_INT_InitPins(TM_SPI_PinsPack_t pinspack) {
		/* Enable SPI pins */
	#if defined(GPIOB)
		if (pinspack == TM_SPI_PinsPack_1) {
			TM_GPIO_InitAlternate(GPIOB, GPIO_PIN_3 | GPIO_PIN_4 |
GPIO_PIN_5, TM_GPIO_OType_PP, TM_GPIO_PuPd_NOPULL, TM_GPIO_Speed_High,
GPIO_AF_SPI3);
		}
	#endif
	#if defined(GPIOC)
		if (pinspack == TM_SPI_PinsPack_2) {
			TM_GPIO_InitAlternate(GPIOC, GPIO_PIN_10 | GPIO_PIN_11
| GPIO_PIN_12, TM_GPIO_OType_PP, TM_GPIO_PuPd_NOPULL,
TM_GPIO_Speed_High, GPIO_AF_SPI3);
		}
	#endif
		if (pinspack == TM_SPI_PinsPack_Custom) {
			/* Call user function */
```

```
			TM_SPI_InitCustomPinsCallback(SPI3, GPIO_AF_SPI3);
		}
	}
	#endif

	#ifdef USE_SPI4
	void TM_SPI4_INT_InitPins(TM_SPI_PinsPack_t pinspack) {
		/* Init SPI pins */
	#if defined(GPIOE)
		if (pinspack == TM_SPI_PinsPack_1) {
			TM_GPIO_InitAlternate(GPIOE, GPIO_PIN_2 | GPIO_PIN_5 |
GPIO_PIN_6, TM_GPIO_OType_PP, TM_GPIO_PuPd_NOPULL, TM_GPIO_Speed_High,
GPIO_AF_SPI4);
		}
	#endif
	#if defined(GPIOE)
		if (pinspack == TM_SPI_PinsPack_2) {
			TM_GPIO_InitAlternate(GPIOE, GPIO_PIN_12 | GPIO_PIN_13
| GPIO_PIN_14, TM_GPIO_OType_PP, TM_GPIO_PuPd_NOPULL,
TM_GPIO_Speed_High, GPIO_AF_SPI4);
		}
	#endif
		if (pinspack == TM_SPI_PinsPack_Custom) {
			/* Call user function */
			TM_SPI_InitCustomPinsCallback(SPI4, GPIO_AF_SPI4);
		}
	}
	#endif

	#ifdef USE_SPI5
	void TM_SPI5_INT_InitPins(TM_SPI_PinsPack_t pinspack) {
		/* Init SPI pins */
	#if defined(GPIOF)
		if (pinspack == TM_SPI_PinsPack_1) {
			TM_GPIO_InitAlternate(GPIOF, GPIO_PIN_7 | GPIO_PIN_8 |
GPIO_PIN_9, TM_GPIO_OType_PP, TM_GPIO_PuPd_NOPULL, TM_GPIO_Speed_High,
GPIO_AF_SPI5);
		}
	#endif
	#if defined(GPIOF) && defined(GPIOH)
		if (pinspack == TM_SPI_PinsPack_2) {
			TM_GPIO_InitAlternate(GPIOF, GPIO_PIN_11,
TM_GPIO_OType_PP, TM_GPIO_PuPd_NOPULL, TM_GPIO_Speed_High,
GPIO_AF_SPI5);
			TM_GPIO_InitAlternate(GPIOH, GPIO_PIN_6 | GPIO_PIN_7,
TM_GPIO_OType_PP, TM_GPIO_PuPd_NOPULL, TM_GPIO_Speed_High,
GPIO_AF_SPI5);
		}
	#endif
		if (pinspack == TM_SPI_PinsPack_Custom) {
			/* Call user function */
			TM_SPI_InitCustomPinsCallback(SPI5, GPIO_AF_SPI5);
```

```
        }
    }
    #endif

    #ifdef USE_SPI6
    void TM_SPI6_INT_InitPins(TM_SPI_PinsPack_t pinspack) {
    #if defined(GPIOG)
        if (pinspack == TM_SPI_PinsPack_1) {
            /* Init SPI pins */
            TM_GPIO_InitAlternate(GPIOG, GPIO_PIN_12 | GPIO_PIN_13
| GPIO_PIN_14, TM_GPIO_OType_PP, TM_GPIO_PuPd_NOPULL,
TM_GPIO_Speed_High, GPIO_AF_SPI6);
        }
    #endif
        if (pinspack == TM_SPI_PinsPack_Custom) {
            /* Call user function */
            TM_SPI_InitCustomPinsCallback(SPI6, GPIO_AF_SPI6);
        }
    }
    #endif
```

A.3.5. tm_stm32f4_usart

```
    /**
     * |----------------------------------------------------------------
---------
     * | Copyright (C) Tilen Majerle, 2014
     * |
     * | This program is free software: you can redistribute it
and/or modify
     * | it under the terms of the GNU General Public License as
published by
     * | the Free Software Foundation, either version 3 of the
License, or
     * | any later version.
     * |
     * | This program is distributed in the hope that it will be
useful,
     * | but WITHOUT ANY WARRANTY; without even the implied warranty
of
     * | MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See
the
     * | GNU General Public License for more details.
     * |
     * | You should have received a copy of the GNU General Public
License
     * | along with this program.  If not, see
<http://www.gnu.org/licenses/>.
     * |----------------------------------------------------------------
---------
     */
```

```
#include "tm_stm32f4_usart.h"

/**
 * @brief Internal USART struct
 */
typedef struct {
     uint8_t *Buffer;
     uint16_t Size;
     uint16_t Num;
     uint16_t In;
     uint16_t Out;
     uint8_t Initialized;
     uint8_t StringDelimiter;
} TM_USART_t;

/* Set variables for buffers */
#ifdef USE_USART1
uint8_t TM_USART1_Buffer[TM_USART1_BUFFER_SIZE];
#endif
#ifdef USE_USART2
uint8_t TM_USART2_Buffer[TM_USART2_BUFFER_SIZE];
#endif
#ifdef USE_USART3
uint8_t TM_USART3_Buffer[TM_USART3_BUFFER_SIZE];
#endif
#ifdef USE_UART4
uint8_t TM_UART4_Buffer[TM_UART4_BUFFER_SIZE];
#endif
#ifdef USE_UART5
uint8_t TM_UART5_Buffer[TM_UART5_BUFFER_SIZE];
#endif
#ifdef USE_USART6
uint8_t TM_USART6_Buffer[TM_USART6_BUFFER_SIZE];
#endif
#ifdef USE_UART7
uint8_t TM_UART7_Buffer[TM_UART7_BUFFER_SIZE];
#endif
#ifdef USE_UART8
uint8_t TM_UART8_Buffer[TM_UART8_BUFFER_SIZE];
#endif

#ifdef USE_USART1
TM_USART_t TM_USART1 = {TM_USART1_Buffer, TM_USART1_BUFFER_SIZE, 
0, 0, 0, 0, USART_STRING_DELIMITER};
#endif
#ifdef USE_USART2
TM_USART_t TM_USART2 = {TM_USART2_Buffer, TM_USART2_BUFFER_SIZE, 
0, 0, 0, 0, USART_STRING_DELIMITER};
#endif
#ifdef USE_USART3
TM_USART_t TM_USART3 = {TM_USART3_Buffer, TM_USART3_BUFFER_SIZE, 
0, 0, 0, 0, USART_STRING_DELIMITER};
```

```
#endif
#ifdef USE_UART4
TM_USART_t TM_UART4 = {TM_UART4_Buffer, TM_UART4_BUFFER_SIZE, 0,
0, 0, 0, USART_STRING_DELIMITER};
#endif
#ifdef USE_UART5
TM_USART_t TM_UART5 = {TM_UART5_Buffer, TM_UART5_BUFFER_SIZE, 0,
0, 0, 0, USART_STRING_DELIMITER};
#endif
#ifdef USE_USART6
TM_USART_t TM_USART6 = {TM_USART6_Buffer, TM_USART6_BUFFER_SIZE,
0, 0, 0, 0, USART_STRING_DELIMITER};
#endif
#ifdef USE_UART7
TM_USART_t TM_UART7 = {TM_UART7_Buffer, TM_UART7_BUFFER_SIZE, 0,
0, 0, 0, USART_STRING_DELIMITER};
#endif
#ifdef USE_UART8
TM_USART_t TM_UART8 = {TM_UART8_Buffer, TM_UART8_BUFFER_SIZE, 0,
0, 0, 0, USART_STRING_DELIMITER};
#endif

/* Private functions */
void TM_USART1_InitPins(TM_USART_PinsPack_t pinspack);
void TM_USART2_InitPins(TM_USART_PinsPack_t pinspack);
void TM_USART3_InitPins(TM_USART_PinsPack_t pinspack);
void TM_UART4_InitPins(TM_USART_PinsPack_t pinspack);
void TM_UART5_InitPins(TM_USART_PinsPack_t pinspack);
void TM_USART6_InitPins(TM_USART_PinsPack_t pinspack);
void TM_UART7_InitPins(TM_USART_PinsPack_t pinspack);
void TM_UART8_InitPins(TM_USART_PinsPack_t pinspack);
void TM_USART_INT_InsertToBuffer(TM_USART_t* u, uint8_t c);
TM_USART_t* TM_USART_INT_GetUsart(USART_TypeDef* USARTx);
uint8_t TM_USART_INT_GetSubPriority(USART_TypeDef* USARTx);
uint8_t TM_USART_BufferFull(USART_TypeDef* USARTx);

/* Private initializator function */
static void TM_USART_INT_Init(
     USART_TypeDef* USARTx,
     TM_USART_PinsPack_t pinspack,
     uint32_t baudrate,
     TM_USART_HardwareFlowControl_t FlowControl,
     uint32_t Mode,
     uint32_t Parity,
     uint32_t StopBits,
     uint32_t WordLength
);

void TM_USART_Init(USART_TypeDef* USARTx, TM_USART_PinsPack_t
pinspack, uint32_t baudrate) {
#ifdef USE_USART1
     if (USARTx == USART1) {
```

```
                TM_USART_INT_Init(USART1, pinspack, baudrate, TM_USART1_HARDWARE_FLOW_CONTROL, TM_USART1_MODE, TM_USART1_PARITY, TM_USART1_STOP_BITS, TM_USART1_WORD_LENGTH);
            }
        #endif
        #ifdef USE_USART2
            if (USARTx == USART2) {
                TM_USART_INT_Init(USART2, pinspack, baudrate, TM_USART2_HARDWARE_FLOW_CONTROL, TM_USART2_MODE, TM_USART2_PARITY, TM_USART2_STOP_BITS, TM_USART2_WORD_LENGTH);
            }
        #endif
        #ifdef USE_USART3
            if (USARTx == USART3) {
                TM_USART_INT_Init(USART3, pinspack, baudrate, TM_USART3_HARDWARE_FLOW_CONTROL, TM_USART3_MODE, TM_USART3_PARITY, TM_USART3_STOP_BITS, TM_USART3_WORD_LENGTH);
            }
        #endif
        #ifdef USE_UART4
            if (USARTx == UART4) {
                TM_USART_INT_Init(UART4, pinspack, baudrate, TM_UART4_HARDWARE_FLOW_CONTROL, TM_UART4_MODE, TM_UART4_PARITY, TM_UART4_STOP_BITS, TM_UART4_WORD_LENGTH);
            }
        #endif
        #ifdef USE_UART5
            if (USARTx == UART5) {
                TM_USART_INT_Init(UART5, pinspack, baudrate, TM_UART5_HARDWARE_FLOW_CONTROL, TM_UART5_MODE, TM_UART5_PARITY, TM_UART5_STOP_BITS, TM_UART5_WORD_LENGTH);
            }
        #endif
        #ifdef USE_USART6
            if (USARTx == USART6) {
                TM_USART_INT_Init(USART6, pinspack, baudrate, TM_USART6_HARDWARE_FLOW_CONTROL, TM_USART6_MODE, TM_USART6_PARITY, TM_USART6_STOP_BITS, TM_USART6_WORD_LENGTH);
            }
        #endif
        #ifdef USE_UART7
            if (USARTx == UART7) {
                TM_USART_INT_Init(UART7, pinspack, baudrate, TM_UART7_HARDWARE_FLOW_CONTROL, TM_UART7_MODE, TM_UART7_PARITY, TM_UART7_STOP_BITS, TM_UART7_WORD_LENGTH);
            }
        #endif
        #ifdef USE_UART8
            if (USARTx == UART8) {
                TM_USART_INT_Init(UART8, pinspack, baudrate, TM_UART8_HARDWARE_FLOW_CONTROL, TM_UART8_MODE, TM_UART8_PARITY, TM_UART8_STOP_BITS, TM_UART8_WORD_LENGTH);
```

```
		}
	#endif
	}

	void TM_USART_InitWithFlowControl(USART_TypeDef* USARTx,
TM_USART_PinsPack_t pinspack, uint32_t baudrate,
TM_USART_HardwareFlowControl_t FlowControl) {
	#ifdef USE_USART1
		if (USARTx == USART1) {
			TM_USART_INT_Init(USART1, pinspack, baudrate,
FlowControl, TM_USART1_MODE, TM_USART1_PARITY, TM_USART1_STOP_BITS,
TM_USART1_WORD_LENGTH);
		}
	#endif
	#ifdef USE_USART2
		if (USARTx == USART2) {
			TM_USART_INT_Init(USART2, pinspack, baudrate,
FlowControl, TM_USART2_MODE, TM_USART2_PARITY, TM_USART2_STOP_BITS,
TM_USART2_WORD_LENGTH);
		}
	#endif
	#ifdef USE_USART3
		if (USARTx == USART3) {
			TM_USART_INT_Init(USART3, pinspack, baudrate,
FlowControl, TM_USART3_MODE, TM_USART3_PARITY, TM_USART3_STOP_BITS,
TM_USART3_WORD_LENGTH);
		}
	#endif
	#ifdef USE_UART4
		if (USARTx == UART4) {
			TM_USART_INT_Init(UART4, pinspack, baudrate,
FlowControl, TM_UART4_MODE, TM_UART4_PARITY, TM_UART4_STOP_BITS,
TM_UART4_WORD_LENGTH);
		}
	#endif
	#ifdef USE_UART5
		if (USARTx == UART5) {
			TM_USART_INT_Init(UART5, pinspack, baudrate,
FlowControl, TM_UART5_MODE, TM_UART5_PARITY, TM_UART5_STOP_BITS,
TM_UART5_WORD_LENGTH);
		}
	#endif
	#ifdef USE_USART6
		if (USARTx == USART6) {
			TM_USART_INT_Init(USART6, pinspack, baudrate,
FlowControl, TM_USART6_MODE, TM_USART6_PARITY, TM_USART6_STOP_BITS,
TM_USART6_WORD_LENGTH);
		}
	#endif
	#ifdef USE_UART7
		if (USARTx == UART7) {
```

```
			TM_USART_INT_Init(UART7, pinspack, baudrate,
FlowControl, TM_UART7_MODE, TM_UART7_PARITY, TM_UART7_STOP_BITS,
TM_UART7_WORD_LENGTH);
		}
	#endif
	#ifdef USE_UART8
		if (USARTx == UART8) {
			TM_USART_INT_Init(UART8, pinspack, baudrate,
FlowControl, TM_UART8_MODE, TM_UART8_PARITY, TM_UART8_STOP_BITS,
TM_UART8_WORD_LENGTH);
		}
	#endif
	}

	uint8_t TM_USART_Getc(USART_TypeDef* USARTx) {
		int8_t c = 0;
		TM_USART_t* u = TM_USART_INT_GetUsart(USARTx);

		/* Check if we have any data in buffer */
		if (u->Num > 0 || u->In != u->Out) {
			/* Check overflow */
			if (u->Out == u->Size) {
				u->Out = 0;
			}

			/* Read character */
			c = u->Buffer[u->Out];

			/* Increase output pointer */
			u->Out++;

			/* Decrease number of elements */
			if (u->Num) {
				u->Num--;
			}
		}

		/* Return character */
		return c;
	}

	uint16_t TM_USART_Gets(USART_TypeDef* USARTx, char* buffer,
uint16_t bufsize) {
		uint16_t i = 0;

		/* Get USART structure */
		TM_USART_t* u = TM_USART_INT_GetUsart(USARTx);

		/* Check for any data on USART */
		if (
			u->Num == 0 ||
/*!< Buffer empty */
```

```
                (
                      !TM_USART_FindCharacter(USARTx, u-
>StringDelimiter) && /*!< String delimiter not in buffer */
                      u->Num != u->Size
/*!< Buffer is not full */
                )
           ) {
                /* Return 0 */
                return 0;
           }

           /* If available buffer size is more than 0 characters */
           while (i < (bufsize - 1)) {
                /* We have available data */
                buffer[i] = (char) TM_USART_Getc(USARTx);

                /* Check for end of string */
                if ((uint8_t) buffer[i] == (uint8_t) u-
>StringDelimiter) {
                      /* Done */
                      break;
                }

                /* Increase */
                i++;
           }

           /* Add zero to the end of string */
           buffer[++i] = 0;

           /* Return number of characters in buffer */
           return i;
      }

      uint8_t TM_USART_BufferEmpty(USART_TypeDef* USARTx) {
           TM_USART_t* u = TM_USART_INT_GetUsart(USARTx);

           /* Check if number of characters is zero in buffer */
           return (u->Num == 0 && u->In == u->Out);
      }

      uint8_t TM_USART_BufferFull(USART_TypeDef* USARTx) {
           TM_USART_t* u = TM_USART_INT_GetUsart(USARTx);

           /* Check if number of characters is the same as buffer size
*/
           return (u->Num == u->Size);
      }

      void TM_USART_ClearBuffer(USART_TypeDef* USARTx) {
           TM_USART_t* u = TM_USART_INT_GetUsart(USARTx);
```

```
        /* Reset variables */
        u->Num = 0;
        u->In = 0;
        u->Out = 0;
    }

    void TM_USART_SetCustomStringEndCharacter(USART_TypeDef* USARTx,
uint8_t Character) {
        /* Get USART structure */
        TM_USART_t* u = TM_USART_INT_GetUsart(USARTx);

        /* Set delimiter */
        u->StringDelimiter = Character;
    }

    uint8_t TM_USART_FindCharacter(USART_TypeDef* USARTx, uint8_t c)
{
        uint16_t num, out;
        TM_USART_t* u = TM_USART_INT_GetUsart(USARTx);

        /* Temp variables */
        num = u->Num;
        out = u->Out;

        while (num > 0) {
            /* Check overflow */
            if (out == u->Size) {
                out = 0;
            }

            /* Check if characters matches */
            if ((uint8_t) u->Buffer[out] == (uint8_t) c) {
                /* Character found */
                return 1;
            }

            /* Set new variables */
            out++;
            num--;
        }

        /* Character is not in buffer */
        return 0;
    }

    void TM_USART_Puts(USART_TypeDef* USARTx, char* str) {
        TM_USART_t* u = TM_USART_INT_GetUsart(USARTx);
        /* If we are not initialized */
        if (u->Initialized == 0) {
            return;
        }
```

```
		/* Go through entire string */
		while (*str) {
			/* Wait to be ready, buffer empty */
			USART_WAIT(USARTx);
			/* Send data */
			USARTx->DR = (uint16_t)(*str++ & 0x01FF);
			/* Wait to be ready, buffer empty */
			USART_WAIT(USARTx);
		}
	}

	void TM_USART_Send(USART_TypeDef* USARTx, uint8_t* DataArray, 
uint16_t count) {
		uint16_t i;
		TM_USART_t* u = TM_USART_INT_GetUsart(USARTx);
		/* If we are not initialized */
		if (u->Initialized == 0) {
			return;
		}

		/* Go through entire data array */
		for (i = 0; i < count; i++) {
			/* Wait to be ready, buffer empty */
			USART_WAIT(USARTx);
			/* Send data */
			USARTx->DR = (uint16_t)(DataArray[i]);
			/* Wait to be ready, buffer empty */
			USART_WAIT(USARTx);
		}
	}

	/* Private functions */
	void TM_USART_INT_InsertToBuffer(TM_USART_t* u, uint8_t c) {
		/* Still available space in buffer */
		if (u->Num < u->Size) {
			/* Check overflow */
			if (u->In == u->Size) {
				u->In = 0;
			}

			/* Add to buffer */
			u->Buffer[u->In] = c;
			u->In++;
			u->Num++;
		}
	}

	__weak void TM_USART_InitCustomPinsCallback(USART_TypeDef* 
USARTx, uint16_t AlternateFunction) {
		/* Custom user function. */
		/* In case user needs functionality for custom pins, this 
function should be declared outside this library */
```

```
}

TM_USART_t* TM_USART_INT_GetUsart(USART_TypeDef* USARTx) {
	TM_USART_t* u;

#ifdef USE_USART1
	if (USARTx == USART1) {
		u = &TM_USART1;
	}
#endif
#ifdef USE_USART2
	if (USARTx == USART2) {
		u = &TM_USART2;
	}
#endif
#ifdef USE_USART3
	if (USARTx == USART3) {
		u = &TM_USART3;
	}
#endif
#ifdef USE_UART4
	if (USARTx == UART4) {
		u = &TM_UART4;
	}
#endif
#ifdef USE_UART5
	if (USARTx == UART5) {
		u = &TM_UART5;
	}
#endif
#ifdef USE_USART6
	if (USARTx == USART6) {
		u = &TM_USART6;
	}
#endif
#ifdef USE_UART7
	if (USARTx == UART7) {
		u = &TM_UART7;
	}
#endif
#ifdef USE_UART8
	if (USARTx == UART8) {
		u = &TM_UART8;
	}
#endif

	return u;
}

uint8_t TM_USART_INT_GetSubPriority(USART_TypeDef* USARTx) {
	uint8_t u;
```

```
	#ifdef USE_USART1
		if (USARTx == USART1) {
			u = 0;
		}
	#endif
	#ifdef USE_USART2
		if (USARTx == USART2) {
			u = 1;
		}
	#endif
	#ifdef USE_USART3
		if (USARTx == USART3) {
			u = 2;
		}
	#endif
	#ifdef USE_UART4
		if (USARTx == UART4) {
			u = 4;
		}
	#endif
	#ifdef USE_UART5
		if (USARTx == UART5) {
			u = 5;
		}
	#endif
	#ifdef USE_USART6
		if (USARTx == USART6) {
			u = 6;
		}
	#endif
	#ifdef USE_UART7
		if (USARTx == UART7) {
			u = 7;
		}
	#endif
	#ifdef USE_UART8
		if (USARTx == UART8) {
			u = 8;
		}
	#endif

		return u;
	}

	#ifdef USE_USART1
	void TM_USART1_InitPins(TM_USART_PinsPack_t pinspack) {
		/* Init pins */
	#if defined(GPIOA)
		if (pinspack == TM_USART_PinsPack_1) {
			TM_GPIO_InitAlternate(GPIOA, GPIO_Pin_9 | GPIO_Pin_10,
TM_GPIO_OType_PP, TM_GPIO_PuPd_UP, TM_GPIO_Speed_High,
GPIO_AF_USART1);
```

```
        }
    #endif
    #if defined(GPIOB)
        if (pinspack == TM_USART_PinsPack_2) {
            TM_GPIO_InitAlternate(GPIOB, GPIO_Pin_6 | GPIO_Pin_7,
TM_GPIO_OType_PP, TM_GPIO_PuPd_UP, TM_GPIO_Speed_High,
GPIO_AF_USART1);
        }
    #endif
        if (pinspack == TM_USART_PinsPack_Custom) {
            /* Init custom pins, callback used */
            TM_USART_InitCustomPinsCallback(USART1,
GPIO_AF_USART1);
        }
    }
    #endif

    #ifdef USE_USART2
    void TM_USART2_InitPins(TM_USART_PinsPack_t pinspack) {
        /* Init pins */
    #if defined(GPIOA)
        if (pinspack == TM_USART_PinsPack_1) {
            TM_GPIO_InitAlternate(GPIOA, GPIO_Pin_2 | GPIO_Pin_3,
TM_GPIO_OType_PP, TM_GPIO_PuPd_UP, TM_GPIO_Speed_High,
GPIO_AF_USART2);
        }
    #endif
    #if defined(GPIOD)
        if (pinspack == TM_USART_PinsPack_2) {
            TM_GPIO_InitAlternate(GPIOD, GPIO_Pin_5 | GPIO_Pin_6,
TM_GPIO_OType_PP, TM_GPIO_PuPd_UP, TM_GPIO_Speed_High,
GPIO_AF_USART2);
        }
    #endif
        if (pinspack == TM_USART_PinsPack_Custom) {
            /* Init custom pins, callback used */
            TM_USART_InitCustomPinsCallback(USART2,
GPIO_AF_USART2);
        }
    }
    #endif

    #ifdef USE_USART3
    void TM_USART3_InitPins(TM_USART_PinsPack_t pinspack) {
        /* Init pins */
    #if defined(GPIOB)
        if (pinspack == TM_USART_PinsPack_1) {
            TM_GPIO_InitAlternate(GPIOB, GPIO_Pin_10 |
GPIO_Pin_11, TM_GPIO_OType_PP, TM_GPIO_PuPd_UP, TM_GPIO_Speed_High,
GPIO_AF_USART3);
        }
    #endif
```

```
	#if defined(GPIOC)
		if (pinspack == TM_USART_PinsPack_2) {
			TM_GPIO_InitAlternate(GPIOC, GPIO_Pin_10 |
GPIO_Pin_11, TM_GPIO_OType_PP, TM_GPIO_PuPd_UP, TM_GPIO_Speed_High,
GPIO_AF_USART3);
		}
	#endif
	#if defined(GPIOD)
		if (pinspack == TM_USART_PinsPack_3) {
			TM_GPIO_InitAlternate(GPIOD, GPIO_Pin_8 | GPIO_Pin_9,
TM_GPIO_OType_PP, TM_GPIO_PuPd_UP, TM_GPIO_Speed_High,
GPIO_AF_USART3);
		}
	#endif
		if (pinspack == TM_USART_PinsPack_Custom) {
			/* Init custom pins, callback used */
			TM_USART_InitCustomPinsCallback(USART3,
GPIO_AF_USART3);
		}
	}
	#endif

	#ifdef USE_UART4
	void TM_UART4_InitPins(TM_USART_PinsPack_t pinspack) {
		/* Init pins */
	#if defined(GPIOA)
		if (pinspack == TM_USART_PinsPack_1) {
			TM_GPIO_InitAlternate(GPIOA, GPIO_Pin_0 | GPIO_Pin_1,
TM_GPIO_OType_PP, TM_GPIO_PuPd_UP, TM_GPIO_Speed_High, GPIO_AF_UART4);
		}
	#endif
	#if defined(GPIOC)
		if (pinspack == TM_USART_PinsPack_2) {
			TM_GPIO_InitAlternate(GPIOC, GPIO_Pin_10 |
GPIO_Pin_11, TM_GPIO_OType_PP, TM_GPIO_PuPd_UP, TM_GPIO_Speed_High,
GPIO_AF_UART4);
		}
	#endif
		if (pinspack == TM_USART_PinsPack_Custom) {
			/* Init custom pins, callback used */
			TM_USART_InitCustomPinsCallback(UART4, GPIO_AF_UART4);
		}
	}
	#endif

	#ifdef USE_UART5
	void TM_UART5_InitPins(TM_USART_PinsPack_t pinspack) {
		/* Init pins */
	#if defined(GPIOC) && defined(GPIOD)
		if (pinspack == TM_USART_PinsPack_1) {
			TM_GPIO_InitAlternate(GPIOC, GPIO_Pin_12,
TM_GPIO_OType_PP, TM_GPIO_PuPd_UP, TM_GPIO_Speed_High, GPIO_AF_UART5);
```

```
                TM_GPIO_InitAlternate(GPIOD, GPIO_Pin_2,
TM_GPIO_OType_PP, TM_GPIO_PuPd_UP, TM_GPIO_Speed_High, GPIO_AF_UART5);
            }
      #endif
            if (pinspack == TM_USART_PinsPack_Custom) {
                  /* Init custom pins, callback used */
                  TM_USART_InitCustomPinsCallback(UART5, GPIO_AF_UART5);
            }
      }
      #endif

      #ifdef USE_USART6
      void TM_USART6_InitPins(TM_USART_PinsPack_t pinspack) {
            /* Init pins */
      #if defined(GPIOC)
            if (pinspack == TM_USART_PinsPack_1) {
                  TM_GPIO_InitAlternate(GPIOC, GPIO_Pin_6 | GPIO_Pin_7,
TM_GPIO_OType_PP, TM_GPIO_PuPd_UP, TM_GPIO_Speed_High,
GPIO_AF_USART6);
            }
      #endif
      #if defined(GPIOG)
            if (pinspack == TM_USART_PinsPack_2) {
                  TM_GPIO_InitAlternate(GPIOG, GPIO_Pin_14 | GPIO_Pin_9,
TM_GPIO_OType_PP, TM_GPIO_PuPd_UP, TM_GPIO_Speed_High,
GPIO_AF_USART6);
            }
      #endif
            if (pinspack == TM_USART_PinsPack_Custom) {
                  /* Init custom pins, callback used */
                  TM_USART_InitCustomPinsCallback(USART6,
GPIO_AF_USART6);
            }
      }
      #endif

      #ifdef USE_UART7
      void TM_UART7_InitPins(TM_USART_PinsPack_t pinspack) {
            /* Init pins */
      #if defined(GPIOE)
            if (pinspack == TM_USART_PinsPack_1) {
                  TM_GPIO_InitAlternate(GPIOE, GPIO_Pin_8 | GPIO_Pin_7,
TM_GPIO_OType_PP, TM_GPIO_PuPd_UP, TM_GPIO_Speed_High, GPIO_AF_UART7);
            }
      #endif
      #if defined(GPIOF)
            if (pinspack == TM_USART_PinsPack_2) {
                  TM_GPIO_InitAlternate(GPIOF, GPIO_Pin_7 | GPIO_Pin_6,
TM_GPIO_OType_PP, TM_GPIO_PuPd_UP, TM_GPIO_Speed_High, GPIO_AF_UART7);
            }
      #endif
            if (pinspack == TM_USART_PinsPack_Custom) {
```

```
			/* Init custom pins, callback used */
			TM_USART_InitCustomPinsCallback(UART7, GPIO_AF_UART7);
		}
	}
	#endif

	#ifdef USE_UART8
	void TM_UART8_InitPins(TM_USART_PinsPack_t pinspack) {
		/* Init pins */
	#if defined(GPIOE)
		if (pinspack == TM_USART_PinsPack_1) {
			TM_GPIO_InitAlternate(GPIOE, GPIO_Pin_1 | GPIO_Pin_0,
TM_GPIO_OType_PP, TM_GPIO_PuPd_UP, TM_GPIO_Speed_High, GPIO_AF_UART8);
		}
	#endif
		if (pinspack == TM_USART_PinsPack_Custom) {
			/* Init custom pins, callback used */
			TM_USART_InitCustomPinsCallback(UART8, GPIO_AF_UART8);
		}
	}
	#endif

	#ifdef USE_USART1
	void USART1_IRQHandler(void) {
		/* Check if interrupt was because data is received */
		if (USART1->SR & USART_SR_RXNE) {
			#ifdef TM_USART1_USE_CUSTOM_IRQ
				/* Call user function */
				TM_USART1_ReceiveHandler(USART1->DR);
			#else
				/* Put received data into internal buffer */
				TM_USART_INT_InsertToBuffer(&TM_USART1, USART1-
>DR);
			#endif
		}
	}
	#endif

	#ifdef USE_USART2
	void USART2_IRQHandler(void) {
		/* Check if interrupt was because data is received */
		if (USART2->SR & USART_SR_RXNE) {
			#ifdef TM_USART2_USE_CUSTOM_IRQ
				/* Call user function */
				TM_USART2_ReceiveHandler(USART2->DR);
			#else
				/* Put received data into internal buffer */
				TM_USART_INT_InsertToBuffer(&TM_USART2, USART2-
>DR);
			#endif
		}
	}
```

```
	#endif

	#ifdef USE_USART3
	void USART3_IRQHandler(void) {
		/* Check if interrupt was because data is received */
		if (USART3->SR & USART_SR_RXNE) {
			#ifdef TM_USART3_USE_CUSTOM_IRQ
				/* Call user function */
				TM_USART3_ReceiveHandler(USART3->DR);
			#else
				/* Put received data into internal buffer */
				TM_USART_INT_InsertToBuffer(&TM_USART3, USART3-
>DR);
			#endif
		}
	}
	#endif

	#ifdef USE_UART4
	void UART4_IRQHandler(void) {
		/* Check if interrupt was because data is received */
		if (UART4->SR & USART_SR_RXNE) {
			#ifdef TM_UART4_USE_CUSTOM_IRQ
				/* Call user function */
				TM_UART4_ReceiveHandler(UART4->DR);
			#else
				/* Put received data into internal buffer */
				TM_USART_INT_InsertToBuffer(&TM_UART4, UART4-
>DR);
			#endif
		}
	}
	#endif

	#ifdef USE_UART5
	void UART5_IRQHandler(void) {
		/* Check if interrupt was because data is received */
		if (UART5->SR & USART_SR_RXNE) {
			#ifdef TM_UART5_USE_CUSTOM_IRQ
				/* Call user function */
				TM_UART5_ReceiveHandler(UART5->DR);
			#else
				/* Put received data into internal buffer */
				TM_USART_INT_InsertToBuffer(&TM_UART5, UART5-
>DR);
			#endif
		}
	}
	#endif

	#ifdef USE_USART6
	void USART6_IRQHandler(void) {
```

```
		/* Check if interrupt was because data is received */
		if (USART6->SR & USART_SR_RXNE) {
			#ifdef TM_USART6_USE_CUSTOM_IRQ
				/* Call user function */
				TM_USART6_ReceiveHandler(USART6->DR);
			#else
				/* Put received data into internal buffer */
				TM_USART_INT_InsertToBuffer(&TM_USART6, USART6-
>DR);
			#endif
		}
	}
	#endif

	#ifdef USE_UART7
	void UART7_IRQHandler(void) {
		/* Check if interrupt was because data is received */
		if (UART7->SR & USART_SR_RXNE) {
			#ifdef TM_UART7_USE_CUSTOM_IRQ
				/* Call user function */
				TM_UART7_ReceiveHandler(UART7->DR);
			#else
				/* Put received data into internal buffer */
				TM_USART_INT_InsertToBuffer(&TM_UART7, UART7-
>DR);
			#endif
		}
	}
	#endif

	#ifdef USE_UART8
	void UART8_IRQHandler(void) {
		/* Check if interrupt was because data is received */
		if (UART8->SR & USART_SR_RXNE) {
			#ifdef TM_UART8_USE_CUSTOM_IRQ
				/* Call user function */
				TM_UART8_ReceiveHandler(UART8->DR);
			#else
				/* Put received data into internal buffer */
				TM_USART_INT_InsertToBuffer(&TM_UART8, UART8-
>DR);
			#endif
		}
	}
	#endif

	static void TM_USART_INT_Init(
		USART_TypeDef* USARTx,
		TM_USART_PinsPack_t pinspack,
		uint32_t baudrate,
		TM_USART_HardwareFlowControl_t FlowControl,
		uint32_t Mode,
```

```
	uint32_t Parity,
	uint32_t StopBits,
	uint32_t WordLength
) {
	USART_InitTypeDef USART_InitStruct;
	NVIC_InitTypeDef NVIC_InitStruct;
	TM_USART_t* u = TM_USART_INT_GetUsart(USARTx);

	/* Set USART baudrate */
	USART_InitStruct.USART_BaudRate = baudrate;

	/*
	 * Initialize USARTx pins
	 * Set channel for USARTx NVIC
	 */
#ifdef USE_USART1
	if (USARTx == USART1) {
		/* Enable USART clock */
		RCC->APB2ENR |= RCC_APB2ENR_USART1EN;

		/* Init pins */
		TM_USART1_InitPins(pinspack);

		/* Set IRQ channel */
		NVIC_InitStruct.NVIC_IRQChannel = USART1_IRQn;
	}
#endif
#ifdef USE_USART2
	if (USARTx == USART2) {
		/* Enable USART clock */
		RCC->APB1ENR |= RCC_APB1ENR_USART2EN;

		/* Init pins */
		TM_USART2_InitPins(pinspack);

		/* Set IRQ channel */
		NVIC_InitStruct.NVIC_IRQChannel = USART2_IRQn;
	}
#endif
#ifdef USE_USART3
	if (USARTx == USART3) {
		/* Enable USART clock */
		RCC->APB1ENR |= RCC_APB1ENR_USART3EN;

		/* Init pins */
		TM_USART3_InitPins(pinspack);

		/* Set IRQ channel */
		NVIC_InitStruct.NVIC_IRQChannel = USART3_IRQn;
	}
#endif
#ifdef USE_UART4
```

```
	if (USARTx == UART4) {
		/* Enable UART clock */
		RCC->APB1ENR |= RCC_APB1ENR_UART4EN;

		/* Init pins */
		TM_UART4_InitPins(pinspack);

		/* Set IRQ channel */
		NVIC_InitStruct.NVIC_IRQChannel = UART4_IRQn;
	}
#endif
#ifdef USE_UART5
	if (USARTx == UART5) {
		/* Enable UART clock */
		RCC->APB1ENR |= RCC_APB1ENR_UART5EN;

		/* Init pins */
		TM_UART5_InitPins(pinspack);

		/* Set IRQ channel */
		NVIC_InitStruct.NVIC_IRQChannel = UART5_IRQn;
	}
#endif
#ifdef USE_USART6
	if (USARTx == USART6) {
		/* Enable UART clock */
		RCC->APB2ENR |= RCC_APB2ENR_USART6EN;

		/* Init pins */
		TM_USART6_InitPins(pinspack);

		/* Set IRQ channel */
		NVIC_InitStruct.NVIC_IRQChannel = USART6_IRQn;
	}
#endif
#ifdef USE_UART7
	if (USARTx == UART7) {
		/* Enable UART clock */
		RCC->APB1ENR |= RCC_APB1ENR_UART7EN;

		/* Init pins */
		TM_UART7_InitPins(pinspack);

		/* Set IRQ channel */
		NVIC_InitStruct.NVIC_IRQChannel = UART7_IRQn;
	}
#endif
#ifdef USE_UART8
	if (USARTx == UART8) {
		/* Enable UART clock */
		RCC->APB1ENR |= RCC_APB1ENR_UART8EN;
```

```
			/* Init pins */
			TM_UART8_InitPins(pinspack);

			/* Set IRQ channel */
			NVIC_InitStruct.NVIC_IRQChannel = UART8_IRQn;
		}
	#endif

		/* Deinit USART, force reset */
		USART_DeInit(USARTx);

		/* Fill NVIC settings */
		NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
		NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority =
USART_NVIC_PRIORITY;
		NVIC_InitStruct.NVIC_IRQChannelSubPriority =
TM_USART_INT_GetSubPriority(USARTx);
		NVIC_Init(&NVIC_InitStruct);

		/* Fill default settings */
		USART_InitStruct.USART_HardwareFlowControl = FlowControl;
		USART_InitStruct.USART_Mode = Mode;
		USART_InitStruct.USART_Parity = Parity;
		USART_InitStruct.USART_StopBits = StopBits;
		USART_InitStruct.USART_WordLength = WordLength;

		/* We are not initialized */
		u->Initialized = 0;

		do {
			volatile uint32_t x = 0xFFF;
			while (x--);
		} while (0);

		/* Init */
		USART_Init(USARTx, &USART_InitStruct);

		/* Enable RX interrupt */
		USARTx->CR1 |= USART_CR1_RXNEIE;

		/* We are initialized now */
		u->Initialized = 1;

		/* Enable USART peripheral */
		USARTx->CR1 |= USART_CR1_UE;
	}
```

A.3.6. tm_stm32f4_usart

```
/**
 * |----------------------------------------------------------------------
---------
 * | Copyright (C) Tilen Majerle, 2014
 * |
 * | This program is free software: you can redistribute it 
and/or modify
 * | it under the terms of the GNU General Public License as 
published by
 * | the Free Software Foundation, either version 3 of the 
License, or
 * | any later version.
 * |
 * | This program is distributed in the hope that it will be 
useful,
 * | but WITHOUT ANY WARRANTY; without even the implied warranty 
of
 * | MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See 
the
 * | GNU General Public License for more details.
 * |
 * | You should have received a copy of the GNU General Public 
License
 * | along with this program.  If not, see 
<http://www.gnu.org/licenses/>.
 * |----------------------------------------------------------------------
---------
 */
#include "tm_stm32f4_usb_vcp.h"
#include "usb_vcp/usbd_usr.h"

/* Private */
uint8_t 
TM_INT_USB_VCP_ReceiveBuffer[USB_VCP_RECEIVE_BUFFER_LENGTH];
uint32_t tm_int_usb_vcp_buf_in, tm_int_usb_vcp_buf_out, 
tm_int_usb_vcp_buf_num;
extern TM_USB_VCP_Result TM_USB_VCP_INT_Status;
extern LINE_CODING linecoding;
uint8_t TM_USB_VCP_INT_Init = 0;
USB_OTG_CORE_HANDLE   USB_OTG_dev;

/* USB VCP Internal receive buffer */
extern uint8_t 
TM_INT_USB_VCP_ReceiveBuffer[USB_VCP_RECEIVE_BUFFER_LENGTH];

TM_USB_VCP_Result TM_USB_VCP_Init(void) {
     /* Initialize USB */
     USBD_Init( &USB_OTG_dev,
#ifdef USE_USB_OTG_FS
```

```
                            USB_OTG_FS_CORE_ID,
#else
                            USB_OTG_HS_CORE_ID,
#endif
                            &USR_desc,
                            &USBD_CDC_cb,
                            &USR_cb);

        /* Reset buffer counters */
        tm_int_usb_vcp_buf_in = 0;
        tm_int_usb_vcp_buf_out = 0;
        tm_int_usb_vcp_buf_num = 0;

        /* Initialized */
        TM_USB_VCP_INT_Init = 1;

        /* Return OK */
        return TM_USB_VCP_OK;
}

uint8_t TM_USB_VCP_BufferEmpty(void) {
        return (tm_int_usb_vcp_buf_num == 0);
}

uint8_t TM_USB_VCP_BufferFull(void) {
        return (tm_int_usb_vcp_buf_num ==
USB_VCP_RECEIVE_BUFFER_LENGTH);
}

uint8_t TM_USB_VCP_FindCharacter(volatile char c) {
        uint16_t num, out;

        /* Temp variables */
        num = tm_int_usb_vcp_buf_num;
        out = tm_int_usb_vcp_buf_out;

        while (num > 0) {
                /* Check overflow */
                if (out == USB_VCP_RECEIVE_BUFFER_LENGTH) {
                        out = 0;
                }
                if (TM_INT_USB_VCP_ReceiveBuffer[out] == c) {
                        /* Character found */
                        return 1;
                }
                out++;
                num--;
        }

        /* Character is not in buffer */
        return 0;
}
```

```
     TM_USB_VCP_Result TM_USB_VCP_Getc(uint8_t* c) {
          /* Any data in buffer */
          if (tm_int_usb_vcp_buf_num > 0) {
               /* Check overflow */
               if (tm_int_usb_vcp_buf_out >=
USB_VCP_RECEIVE_BUFFER_LENGTH) {
                    tm_int_usb_vcp_buf_out = 0;
               }
               *c =
TM_INT_USB_VCP_ReceiveBuffer[tm_int_usb_vcp_buf_out];
               TM_INT_USB_VCP_ReceiveBuffer[tm_int_usb_vcp_buf_out] =
0;

               /* Set counters */
               tm_int_usb_vcp_buf_out++;
               tm_int_usb_vcp_buf_num--;

               /* Data OK */
               return TM_USB_VCP_DATA_OK;
          }
          *c = 0;
          /* Data not ready */
          return TM_USB_VCP_DATA_EMPTY;
     }

     TM_USB_VCP_Result TM_USB_VCP_Putc(volatile char c) {
          uint8_t ce = (uint8_t)c;

          /* Send data over USB */
          VCP_DataTx(&ce, 1);

          /* Return OK */
          return TM_USB_VCP_OK;
     }

     TM_USB_VCP_Result TM_USB_VCP_Puts(char* str) {
          while (*str) {
               TM_USB_VCP_Putc(*str++);
          }

          /* Return OK */
          return TM_USB_VCP_OK;
     }

     TM_USB_VCP_Result TM_USB_VCP_Send(uint8_t* DataArray, uint32_t
Length) {
          /* Send array */
          VCP_DataTx(DataArray, Length);

          /* Return OK */
          return TM_USB_VCP_OK;
```

```
      }

      uint16_t TM_USB_VCP_Gets(char* buffer, uint16_t bufsize) {
            uint16_t i = 0;
            uint8_t c;

            /* Check for any data on USART */
            if (TM_USB_VCP_BufferEmpty() ||
(!TM_USB_VCP_FindCharacter('\n') && !TM_USB_VCP_BufferFull())) {
                  return 0;
            }

            /* If available buffer size is more than 0 characters */
            while (i < (bufsize - 1)) {
                  /* We have available data */
                  while (TM_USB_VCP_Getc(&c) != TM_USB_VCP_DATA_OK);
                  /* Save new data */
                  buffer[i] = (char) c;
                  /* Check for end of string */
                  if (buffer[i] == '\n') {
                        i++;
                        /* Done */
                        break;
                  } else {
                        i++;
                  }
            }

            /* Add zero to the end of string */
            buffer[i] = 0;

            /* Return number of characters in string */
            return i;
      }

      TM_USB_VCP_Result TM_INT_USB_VCP_AddReceived(uint8_t c) {
            /* Still available data in buffer */
            if (tm_int_usb_vcp_buf_num < USB_VCP_RECEIVE_BUFFER_LENGTH)
{
                  /* Check for overflow */
                  if (tm_int_usb_vcp_buf_in >=
USB_VCP_RECEIVE_BUFFER_LENGTH) {
                        tm_int_usb_vcp_buf_in = 0;
                  }
                  /* Add character to buffer */
                  TM_INT_USB_VCP_ReceiveBuffer[tm_int_usb_vcp_buf_in] =
c;
                  /* Increase counters */
                  tm_int_usb_vcp_buf_in++;
                  tm_int_usb_vcp_buf_num++;

                  /* Return OK */
```

```
            return TM_USB_VCP_OK;
        }

        /* Return Buffer full */
        return TM_USB_VCP_RECEIVE_BUFFER_FULL;
    }

    TM_USB_VCP_Result TM_USB_VCP_GetStatus(void) {
        if (TM_USB_VCP_INT_Init) {
            return TM_USB_VCP_INT_Status;
        }
        return TM_USB_VCP_ERROR;
    }

    TM_USB_VCP_Result TM_USB_VCP_GetSettings(TM_USB_VCP_Settings_t*
Settings) {
        /* Fill data */
        Settings->Baudrate = linecoding.bitrate;
        Settings->DataBits = linecoding.datatype;
        Settings->Parity = linecoding.paritytype;
        Settings->Stopbits = linecoding.format;
        Settings->Changed = linecoding.changed;

        /* Clear changed flag */
        linecoding.changed = 0;

        /* Return OK */
        return TM_USB_VCP_OK;
    }
```