5-1-2018

Design and Manufacturing of a High Performance Indoor Quadcopter

Guangyi Liu
Lehigh University, bitcax1u@hotmail.com

Design and Manufacturing of

a High Performance Indoor Quadcopter

by

Guangyi Liu

A Thesis

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of Master of Science

in Mechanical Engineering

Lehigh University

May 2018

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science.

_____

Date

_____

Nader Motee, Advisor

_____

D. Gary Harlow, Chairperson

Department of Mechanical Engineering and Mechanics

Contents

List of figures

List of tables

Acknowledgements

First of all, I would like to extend my sincere gratitude to my advisor Prof. Nader Motee, for having me in the research group and offering me the opportunity to complete my thesis. His instructive advice and useful suggestions helped me get over all the obstacles I have met when I was writing the thesis. I am deeply grateful of his help during my pursuit of my master's degree.

I would also like to thank Yaser Ghaedsharaf, Arash Amini, Milad Habibi, Alan Bebout, Komel Merchant and Dr. Babak Shirmohammadi for helping me through the entire research and providing great ideas.

Finally, I would like to thank my family for supporting me throughout my education and having great confidence in me all through these years.

Abstract

In DCDS (Distributed Control and Dynamical Systems) laboratory, we are using quadcopter flying inside the building with motion capture system. In the quadcopter base, we are designing a new flight controller which has a high-speed processor and fast Wi-Fi communication. And the price is affordable in case of crashes during the research.

The IMU used on the quadcopter is MPU6050 and HMC5883L magnetometer, they provide accurate Euler angle, angular velocity and heading at 200Hz. The controller used for attitude is cascade PID controller which controls the angle and the angular velocity at the same time, it's robust and reliable. And the controller used for altitude is a PID controller with the feedback from sonar.

The new quadcopter can perform a stable hover, safe takeoff and landing, it's currently not as good as commercial flight controllers in flight performance, but it has a great potential of modification and capable of handling a lot of future additional task.

With the feedback form Motion Capture system in the future, we will have a more accurate heading and altitude. Motion Capture system will recognize the markers attached on the frame and send the position and configuration of the object back to the ground station. By measuring the relative distance from markers to cameras, position and configuration information will be more accurate which makes the performance better.

1 Introduction and related work

1.1 Background

The drone testing base is equipped with a high-precision motion capture system, a central server, a wireless communication network, and custom software tools. Our team is developing design methodologies to optimize drone size, flight controller and on-board sensors by integrating and co-designing control and navigation algorithms. As the demand of the research, we need to build a quadcopter which has a powerful processor, support Wi-Fi communication and the cost of itself is not too much.

As shown in the Figure 1.1, our quadcopter is recognized by the Motion Capture system with markers attached on the frame. Motion Capture will send the flight information (altitude, velocity, etc.) to the server, and the server will spread it out with Wi-Fi. The quadcopter will receive the information through a Wi-Fi chip called ESP8266. And the information from Motion Capture system will be sent to the flight controller and then feed to the controller.

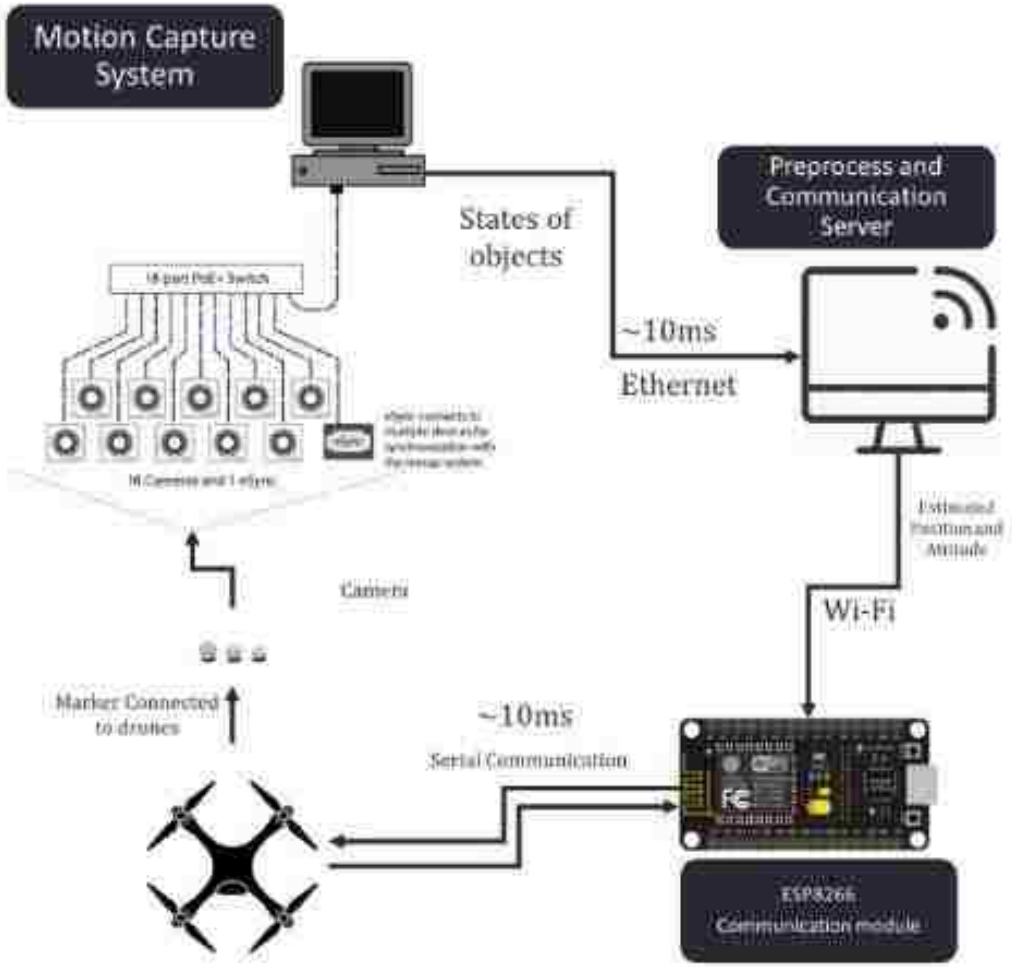


Figure 1.1 The schematic of our current project

1.2 Flight controllers

We started with Cheerson CX-20 quadcopter for a better understanding of how the system of a quadcopter looks like. It has a stable outdoor flight performance, but it is not suitable for the indoor flight for research and it doesn't support the communication with the Motion Capture system for the future tasks.

We then switched to APM flight controller which runs Arducopter, an opensource flight control firmware onboard. The hardware onboard allows quadcopter to

communicate with the ground station or radio controller. The processor inside is Atmel's ATMEGA2560 (16MHz), this processor it not capable of handling too much tasks and carrying Wi-Fi communication.

Figure 1.2 CX-20 quadcopter (left) and APM 2.8 flight controller (right)

The next choice is Pixhawk flight controller which share the same firmware Arducopter with APM but it has a more powerful processor and a SPI/I2C communication port. Pixhawk doesn't have the Wi-Fi module imbedded inside so our solution is connecting it with a Wi-Fi router Espressif ESP8266. This module has a Wi-Fi chip onboard and it can communicate with Pixhawk through SPI (Serial Peripheral Interface Bus) port. But the serial SPI communication has a delay of 10ms for the communication loop which will affect the quality of the controller.

Figure 1.3 Pixhawk (left) and Pixhawk2 (right)

We then decided to choose Espressif ESP32 as a flight controller which was released on September 6, 2016, it has a due core processor and the Wi-Fi module is imbedded within the chip. This substitution will cut down the delay of the serial SPI communication.

Figure 1.4 Espressif ESP32 core chip

ESP32 is capable of functioning reliably in industrial environments, with an operating temperature ranging from -40°C to +125°C. Powered by advanced

calibration circuitries, ESP32 can dynamically remove external circuit imperfections and adapt to changes in external conditions. This feature will ensure the reliable performance of our quadcopter in the extreme conditions.

The most applications of ESP32 is about Internet of Things (IoT) and smart home devices for its excellent Wi-Fi function, and it only has the processor and Wi-Fi chip onboard without any inertial measurement unit (IMU). We will need to attach external IMU to the ESP32 to build the quadcopter.

As shown in Table 1.1, the price of a ESP32 chip is acceptable for research use and it is affordable for us if there is any hardware lost during the quadcopter crash. It's 1/24 of the price of the newest Pixhawk 2.1 and has a better processor than it.

Name of the flight controller	CX-20 (including the whole quadcopter)	Ardupilot APM 2.8	RadioLink Pixhawk PX4	Pixhawk 2.1 (The Cube)	Espressif ESP32
Market price	$165.50	$29.99	$109.99	$238.00	$9.95

Table 1.1 Market price of different flight controllers

1.3 IMUs

There are a lot of IMUs available in the market and two low price IMU came into our sight. One is MPU 6050 and another is MPU6500, they are both from InvenSense company, containing a 3-axis accelerometer and a 3-axis gyroscope, but MPU6500 allows a 32kHz gyro sampling rate compares to the 8kHz sampling rate of MPU6050. In the practice, the high sampling rate of MPU6500 makes it extreme sensitive to the small vibration of the quadcopter frame. Since the motor is always spinning during the flight, all the unneeded vibration will be detected by MPU6500. Additional filter will be acquired for MPU6500 to filter out the redundant data, and there will be a high possibility that this filter will cause a remarkable delay to the data. MPU6050 is just the half of the price of the MPU6500 and the 8kHz sampling rate is fast enough for indoor quadcopter research.

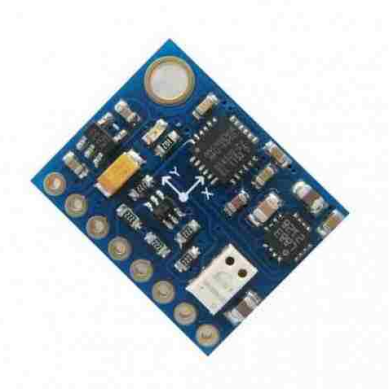


Figure 1.5 GY-87 developing board with MPU6050

1.4 Frames

We started with F450 quadcopter frame with the diagonal length (motor to motor) of 450mm and equipped with 178mm propellers, this one can generate the total thrust of 2040g on full throttle. The scale of this frame will drain the battery too fast and the propeller of this size may cause serious injury when crashed.

We then switched to F330 with the diagonal length (motor to motor) of 330mm and equipped with propellers of 130mm. This frame will ensure a longer battery life and a safer testing environment for research members.

Figure 1.6 F330 quadcopter frame

1.5 Motors and ESCs

The motor we are using is Turnigy L2210A-1650 Brushless Motor, with the max power of 180W and 1650 rpm per Volt. The specifications can be found in Appendix C.

The speed of the brushless DC motor can be controlled by adjusting the timing of pulses of current delivered to the several windings of the motor, but the flight controller can only generate the PWM (pulse-width modulation) signals. The ESC (Electronic speed control) will receive the PWM signals from flight controller and then transform them into pulses of current and control the speed of the motor.

Figure 1.7 Turnigy L2210A-1650 Brushless Motor

Figure 1.8 Turnigy MultiStar 32bit ESC

The ESC unit we are using is Turnigy MultiStar 32bit ESC which support a 480Hz refresh rates and 20A current. This fits well with the motor we have selected. The specifications can be found in Appendix D.

1.6 The total price of quadcopter

Table 1.2 is showing the price specifications about different quadcopters, the total price of the quadcopter can be reduced to $100 if using ESP32. It keeps the same IMU but has a faster processor and Wi-Fi communication.

Name of the flight controller	CX-20 (including the whole quadcopter)	Ardupilot APM 2.8 ($29.99)	RadioLink Pixhawk PX4 ($109.99)	Pixhawk 2.1 ($238.00)	Espressif ESP32 ($9.95)
Processor	32bit microcontroller	Atmel's ATMEGA2560	168 MHz Cortex M4F CPU	32-bit ARM Cortex M4 core with FPU	240 MHz Xtensa dual-core 32-bit LX6 microprocessor
IMU	MPU6050 (included)	MPU6050 (included)	MPU6050 (included)	MPU6050 (included)	MPU6050 ($10.41)
ESC	Included	MultiStar 32bit ESC ($12.03) *4	MultiStar 32bit ESC ($12.03) *4	MultiStar 32bit ESC ($12.03) *4	MultiStar 32bit ESC ($12.03) *4
Motor	Included	L2210A-1650 Brushless Motor ($12.00) * 4	L2210A-1650 Brushless Motor ($12.00) * 4	L2210A-1650 Brushless Motor ($12.00) * 4	L2210A-1650 Brushless Motor ($12.00) * 4

Wi-Fi Module	Not supported	Not supported	ESP 8266 ($6.00)	ESP 8266 ($6.00)	Included
Frame	Included	F330 ($10.22)	F330 ($10.22)	F330 ($10.22)	F330 ($10.22)
Total price	$165.50	$126.11	$212.11	$340.12	$106.07

Table 1.2 Build specifications and total price of different quadcopters

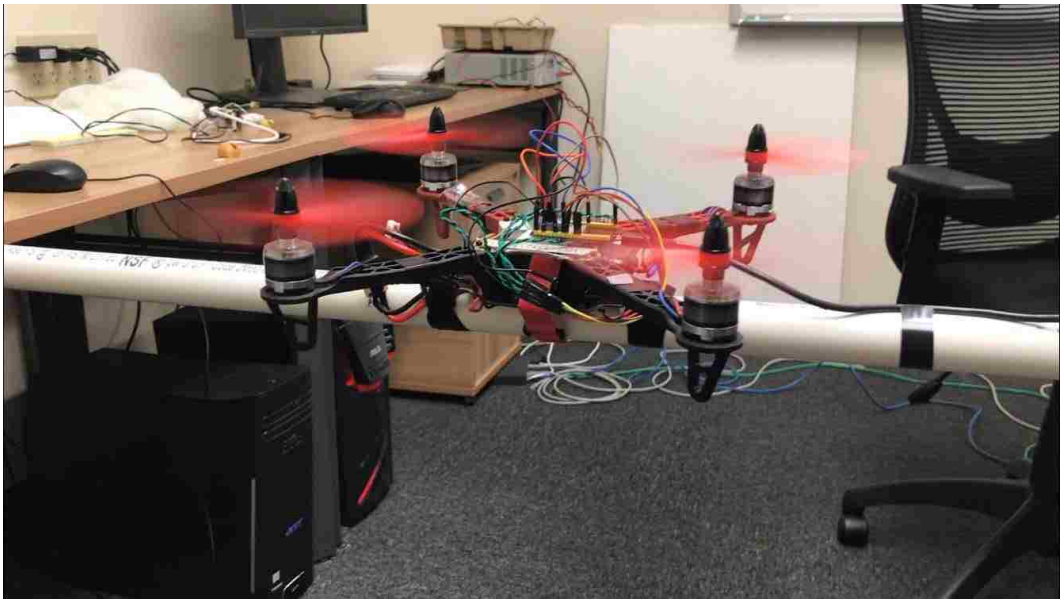


Figure 1.9 Prototype of the new quadcopter

2 Manufacturing the quadcopter

2.1 Programming ESP

The development board of ESP32 (Figure 2.1) is available in the market, it can be programmed by Arduino IDE and the support of the hardware is opensource. The Arduino library supports the full function of Wi-Fi and the PWM generating in ESP32. Power supply for ESP32 can be get from the power module which is connected with the LiPo battery. And power for MPU6050 will be supplied from ESP32.

The pin usage can be specified in Table 2.1, and the pin map of ESP32 can be found in Figure 2.2:

Name of the component	The pin used by the component
GY-87 (MPU6050)	GPIO 21, GPIO22, 3.3V and GND
4 ESCs	GPIO 16~ GPIO 19 and GND
Power Module	5V and GND

Table 2.1 Pin usage of ESP32

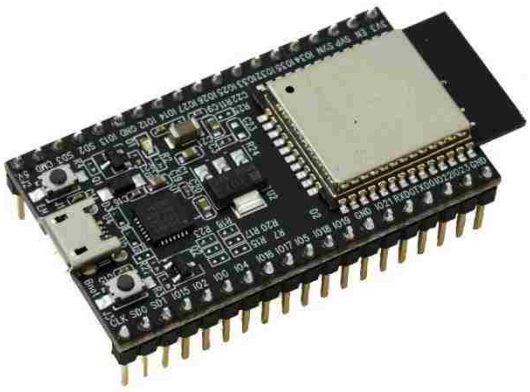


Figure 2.1 Development board of ESP32

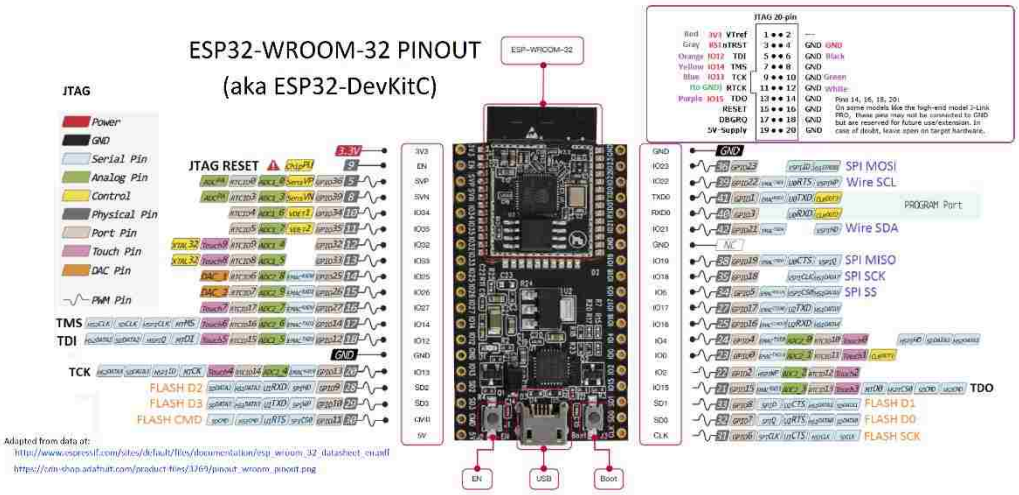


Figure 2.2 The pin map of ESP32

2.2 Connecting MPU with ESP32

2.2.1 Get the raw reading from MPU

The MPU6050 is designed to communicate with the processor through I2C Bus, which needs a SCL pin and a SDA pin connected with the I2C ports of ESP32 which is GPIO port 21 and 22.

The library of MPU6050 and I2C interface is designed by (name of the guy) and it is based on regular Arduino board. Since the I2C structure of ESP32 is different from the regular Arduino board, we need to remove TWBR=24 in the code to allow the function of I2C bus.

The basic function of a MPU6050 is sending back raw readings from gyroscope and accelerometer. These reading could be very noisy and unstable when MPU is not calibrated.

2.2.2 Calibrating MPU

The basic idea of calibrating MPU6050 is to specify the position of the horizontal plane and tell the processor which plane is horizontal and what value for gyroscope raw reading should be 0. The algorithm can be found in Figure 2.3.

First station the MPU on a horizontal surface with z axis point up and perpendicular to the plane. Exclude all potential source of vibration and movement of the plane.

The ideal reading at this situation will be zero for all 3-axis of gyroscope, zero for x, y axis of accelerometer and 1g for z axis of accelerometer since z axis is sensing the gravity. Gather adequate raw data (at least 2000 sets of data) from all axis and calculate a mean value of those data for each axis. These mean values are offsets of each axis accordingly.

For z axis of accelerometer, the offset should be mean value minus 1g. Store all the offsets in the flight controller program and send them to MPU6050 during the initialization of it. Thus, MPU6050 will apply these offsets before send the data back to flight controller. This calibration needed to be done occasionally to make sure the data is accurate.

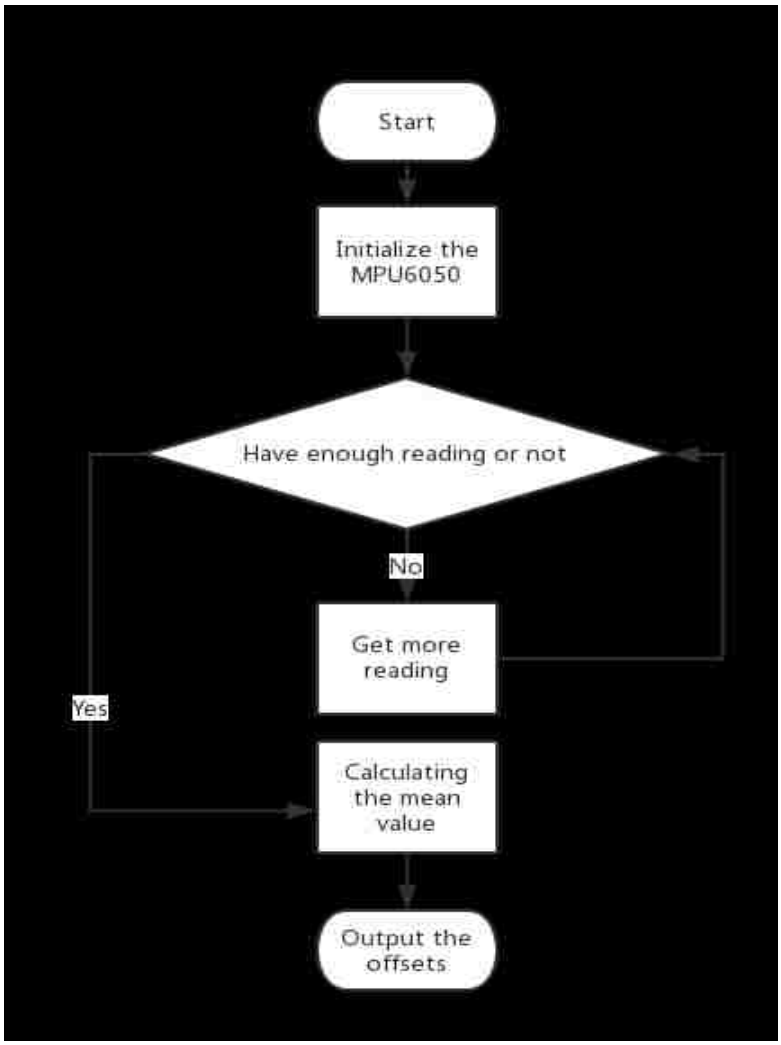


Figure 2.3 Algorithm of MPU calibration

2.2.3 Digital Motion Processor

The raw data are still noisy even after the calibration, the rotation of the motors will create vibrations to the frame and it will lead to a noisy data. Some filter is needed to be applied to the data to have a more accurate reading.

The raw data from gyroscope will be the angular rate of each axis and the current Euler angle will be the integration of the changing rate. The relation between gyroscope reading and angular velocity can be defined as following:

$$gyro_x = p + ge_x, \qquad gyro_y = q + ge_y, \qquad gyro_z = r + ge_z$$

$$\phi_{gyro} = \int gyro_x = \int p + \int ge_x$$

As shown in the equation above, the gyroscope reading are the summation of actual angular velocity and sensor error. We can take roll angle for example, the integral of $gyro_x$ will be the roll angle as the output from gyroscope. Since the existence of ge_x , the error will accumulate during the integration and it will not be canceled by any means. This will make the roll angle reading from MPU6050 continuing to increase or decrease even quadcopter is stationed on the ground. This drifting will happen to all 3-axis.

2.2.3.1 How accelerometer works

Accelerometer is sensing the accelerations on 3-axis, the equation can be defined as following:

$$accel_x = atrue_x + ae_x, \qquad accel_y = atrue_y + ae_y,$$

$$accel_z = atrue_z + ae_z$$

$ature_i$ is actual acceleration on ith axis, ae_i is error of sensor on ith axis and $accel_i$ is reading of accelerometer on ith axis. Since $ae_i \ll atrue_i$ and errors from accelerometer will not accumulate with time, readings of accelerometer can be regard as reliable and accurate.

With the readings from accelerometer, the drift of gyroscope can be compensated easily for pitch and roll angle. We can calculate pitch and roll angles from accelerometer reading when there are no other accelerations than the gravity, and this will not be drifting with time. Accelerometer will provide a more accurate angle reading when quadcopter is not rotating. Gyroscope will provide a more accurate angular velocity reading when quadcopter is rotating since the error of angular velocity is not accumulating with time. A kind of data fusion is needed for pitch and roll angle. The basic idea is trust the gyroscope more for the angular velocity and trust accel more when there is no more movement.

But the accelerometer can't provide any information about yaw angle since the direction of the gravity is parallel to the direction of z-axis, we will need the magnetometer or Motion Capture system to compensate the drift of yaw.

After compensated the drift, the raw data are still noisy since the vibration of the frame will still create errors to the readings. Then we will need a filter applied to raw readings to get the accurate values.

2.2.3.2 Using the DMP

InvenSense has also built a Digital Motion Processor (DMP) inside the MPU6050 and released it to the public. DMP will fuse all the data together as an output of quaternions or Euler angles. It will support an output at a sampling rate of 200Hz. DMP will not lay any burden on the main processor and it saves remarkable time of transforming raw gyro and accel data in to quaternion or Euler angles. The pitch and roll angle output of DMP is accurate and stable but the yaw output is still drifting since there is no standard for yaw. DMP doesn't fuse magnetometer data with yaw readings and this fusion must be done by the processor.

2.2.3.3 Get rid of interruption

DMP data is also transported through I2C bus, and it will pull the interrupt pin to HIGH on the MPU6050 to inform the mainboard what buffer size of the data is. This is not stable and some time it will give out FIFO overflow. Thus, we let the program to measure the buffer size and decide when to cut the buffer and read the data.

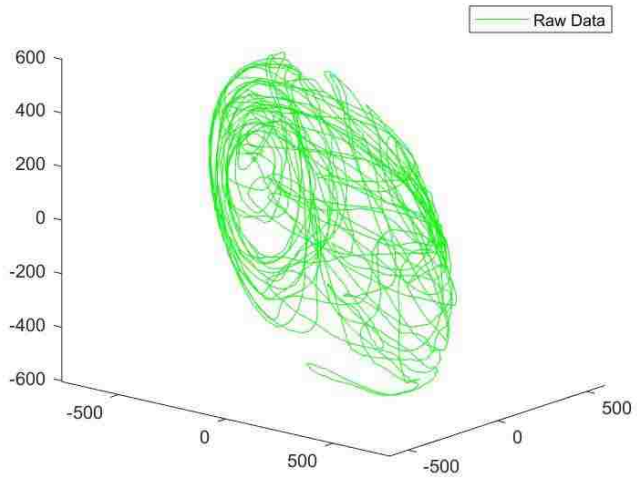
2.2.4 Magnetometer Calibration

The magnetometer will sense the strength of the earth's magnetic field and generate a vector which is pointing the current heading direction based on the reading on

each axis. And this vector will be distorted by hard-iron effect, soft-iron effect and magnetic declination.

The Hard-iron distortion is generated by materials which can produce the constant magnetic field to the environment such as permanent magnet. The magnetic field it generated will be added to the earth's magnetic field and the value of each axis from magnetometer will be added with a constant value. Once the orientation of the hard-iron source is fixed, the additional magnetic field will also be constant. Hard-iron distortion will make the magnetic sphere away from the origin of coordinates. Cellphones, laptops, and the current on the quadcopter would be the origin of hard-iron distortion.

The soft-iron distortion is the result of material that does not generate the magnetic field. This kind of material will influence the surrounding magnetic field and distort the orientation of the reading of the electronic compass. The soft-iron distortion



will make the set of magnetic reading vectors from a sphere to an ellipsoid, and the reading will not be accurate. It will be more common in the indoor environment since all the cables and metal structures will be the origin of the soft-iron distortion.

Figure 2.4 Raw magnetometer reading before calibration

As shown in Figure 2.4, the raw magnetic vector set is not forming a sphere due to the hard-iron and soft-iron distortion.

Magnetic declination is the angle on the horizontal plane between magnetic north and true north. The heading from magnetometer will show the angle between the head and magnetic north if magnetic declination is not applied.

The general procedure of calibrating an electronic compass will be as following:

Point the magnetometer to every direction as much as possible to form a sphere of the reading. The reading vector should form an ellipsoid with long axis pointing random direction and the center is out of origin of coordinates. Our calibration program will find the approximate starting and ending point of the long axis, and the coordinate of middle point of the long axis will be the offset value of the hard-iron distortion. After applying this offset to the raw data, the ellipsoid will be located at the origin of coordinates.

We need to rotate the ellipsoid back to the right orientation and then perform the scaling. The rotation matrix can be defined by the z axis vector and the long axis

vector, and the scaling will be applied to each axis parameter according to the ellipsoid function.

And the scaling procedure can be defined as an optimize problem:

$$\min \sum_{i=1}^{n} (r_i - d)^2$$

$$s.t. \quad r_i = \sqrt{(x_i^2 + y_i^2 + z_i^2)}$$

r_i is the ith mode of the magnetic vector, x, y, z are coordinate of ith magnetic vector, n is the total number of the magnetic vectors.

We can solve the problem with Matlab, and the result is shown in Figure 2.5. It's obvious that the set of vectors has finally form a sphere.

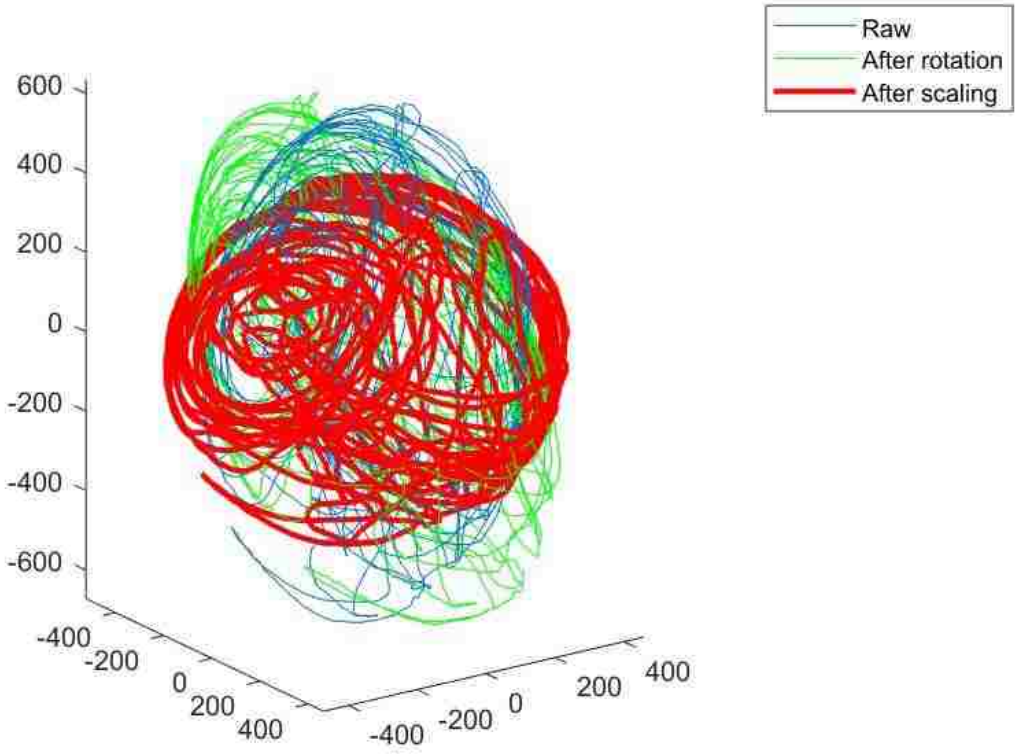


Figure 2.5 The result of scaling

The magnetic declination value can be found on the website of National Centers for Environmental Information (https://www.ngdc.noaa.gov/), it is 12° 9' W at Allentown, PA.

After that the program will print out the rotation matrix and the scaling parameters for the quadcopter. And every magnetic data will be processed with offset first and then rotate, scale and rotate back. Then apply magnetic declination, the output of the magnetometer will be the actual heading of the sensor.

2.3 Wi-Fi Connection

We are using Wi-Fi for communication with the quadcopter, it will take the following tasks:

Receiving Tasks	Sending Tasks
• Commands from ground stations • Flight coefficients from ground station • Flight information from Motion Capture system	• Flight status to ground station

Table 2.2 Tasks of Wi-Fi

TCP and UDP are both available for the communication but TCP is using more steps of communication and this will cause remarkable delay for the response loop of the quadcopter.

We tested the different protocol with measuring the response time for sending a 32 Bytes packet to ESP32 and send back a 32 Bytes packet from ESP32 when the first packet arrives. As shown in Figure 2.6, TCP is using 2 times of time as much as UDP.

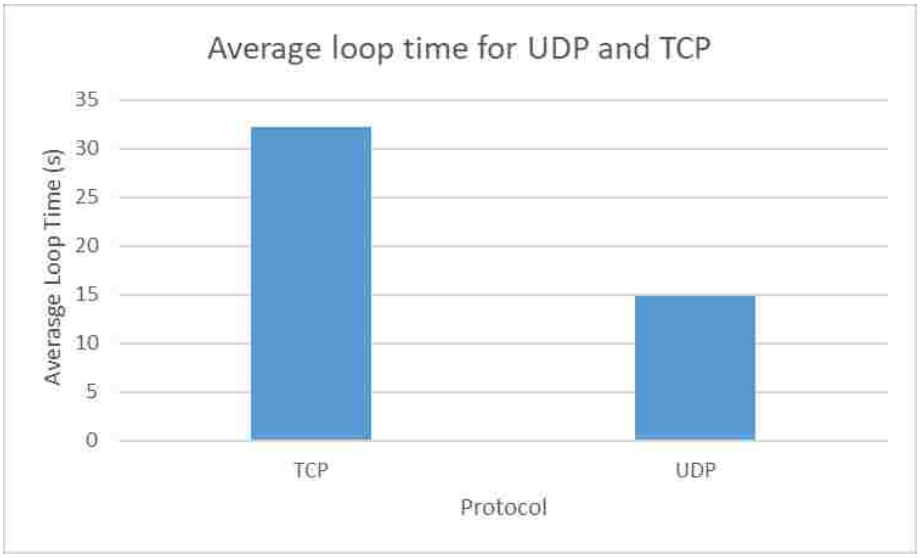


Figure 2.6 Average loop time for UDP and TCP

The structure of the communication is shown in Figure 2.7. There will be a Wi-Fi router at 2.4GHz in the base and both ESP32 and ground station will start a UDP server under the Wi-Fi network, UDP server on the quadcopter and ground station will have distinct IP addresses and same port number. Every packet been sent will have a destination to certain IP address and port number. Thus, they could communicate under the same Wi-Fi network as UDP servers.

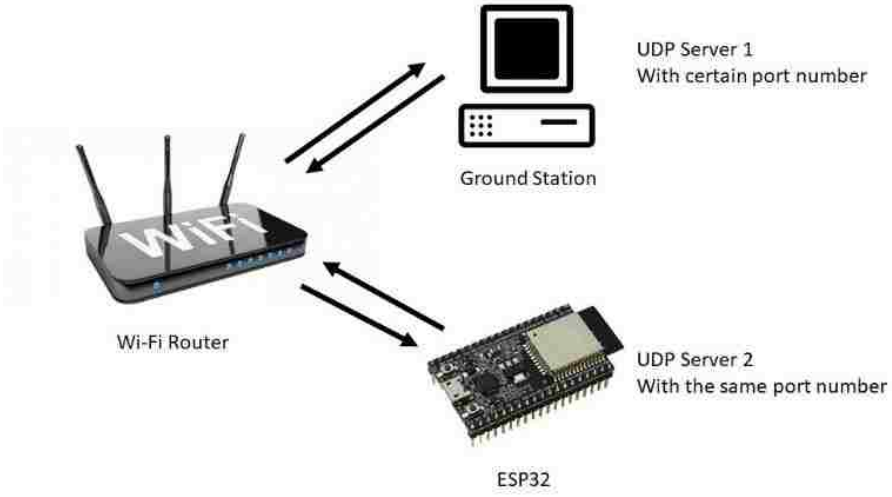


Figure 2.7 The Wi-Fi communication

2.4 Calibrating the ESCs

The ESP32 can generate PWM signals for all GPIO pins on the board, the quadcopter will need 4 pins to send 4 different PWM signals to ESCs. The ESC will enter calibration mode if the throttle is set to max before connected to the power. And after that it will recognize the next PWM signal as the zero throttle. We set the full throttle as 900 PWM and the zero throttle is 500PWM, this throttle resolution is good enough for the quadcopter motors.

This calibration needed to be done before the first run, and we let the processor to send zero throttle before the flight to make sure the ESC will enter the correct mode.

3 The model of the drone

Quadcopter can perform a 6 degrees of freedom motion by adjusting the rotation speed of four motors, we can set the coordinates as shown in the figure, all the motions can be classified as moving along x, y, z axis, rotating around x, y, z axis (also defined as roll, pitch, and yaw). The definitions of axis and Euler angles is shown in Figure 3.1.

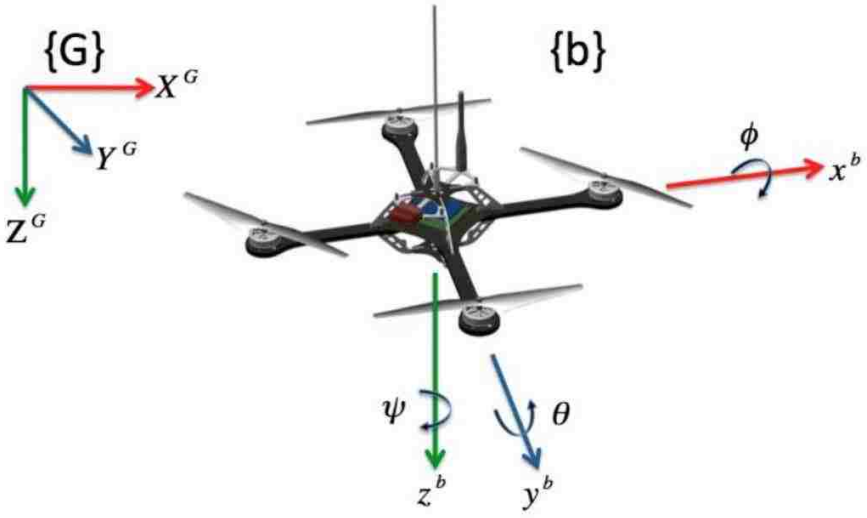


Figure 3.1 Definitions of coordinates

The roll, pitch, yaw and altitude changing motion can be accomplished as shown in Figure 3.2.

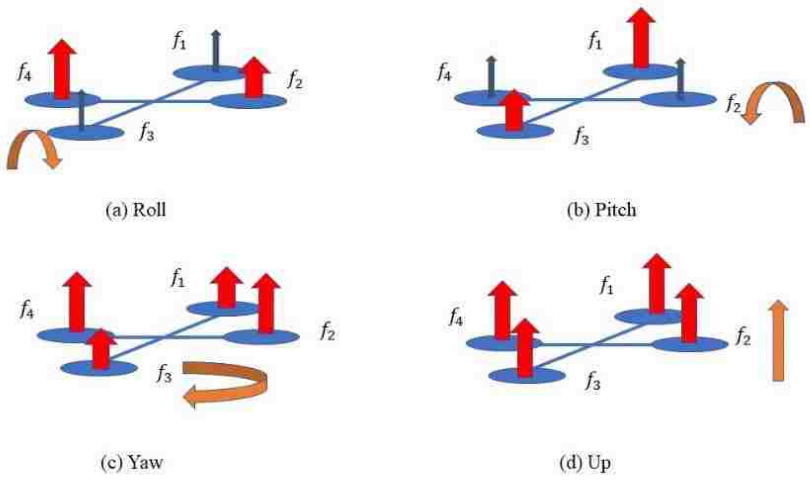


Figure 3.2 Changing motion with motor speed

To build the model of the quadcopter, we are going to make following assumptions first:

a) Assume the quadcopter as a rigid body, ignore all the deformation of the frame during the flight.
b) The shape and mass distribution of quadcopter are symmetric.
c) Ignore the ground effect, vibration of the motor and the influence of air flows between each propeller.
d) The mass and the inertia doesn't change during the flight.

And the rotation matrix will be needed the ground coordinates and the body coordinates of the quadcopter. It can be defined as following:

$$a^B = R_G^b A^G = R(\phi)R(\theta)R(\varphi)A^G$$

$$R(\varphi) = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, R(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix},$$

$$R(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}$$

R_G^b

$$= \begin{bmatrix} \cos(\psi)\cos(\theta) & \sin(\psi)\cos(\theta) & -\sin(\theta) \\ \cos(\psi)\sin(\phi)\sin(\theta) - \cos(\phi)\sin(\psi) & \sin(\phi)\sin(\psi)\sin(\theta) + \cos(\psi)\cos(\phi) & \cos(\theta)\sin(\phi) \\ \cos(\phi)\cos(\psi)\sin(\theta) + \sin(\phi)\sin(\psi) & \cos(\phi)\sin(\psi)\sin(\theta) - \cos(\psi)\sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix}$$

$$A^G = R_b^G a^b = R(\phi)^T R(\theta)^T R(\varphi)^T a^b = \left(R_G^b\right)^T a^b$$

R_b^G

$$= \begin{bmatrix} \cos(\varphi)\cos(\theta) & \cos(\varphi)\sin(\phi)\sin(\theta) - \cos(\phi)\sin(\varphi) & \cos(\phi)\cos(\varphi)\sin(\theta) + \sin(\phi)\sin(\varphi) \\ \sin(\varphi)\cos(\theta) & \sin(\phi)\sin(\varphi)\sin(\theta) + \cos(\varphi)\cos(\phi) & \cos(\phi)\sin(\varphi)\sin(\theta) - \cos(\varphi)\sin(\phi) \\ -\sin(\theta) & \cos(\theta)\sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix}$$

a^b is body coordinate, A^G is ground coordinate, R_G^b is the rotation matrix from ground coordinate to body coordinate, R_b^G is the rotation matrix from body coordinate to ground coordinate.

Since the angular velocity is defined in the body coordinates and the Euler is defined in the ground coordinates, we can derive the transformation between the angular velocity and the Euler angle rate as following:

$$\omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = R(\phi)R(\theta)\begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + R(\phi)\begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix}$$

$$\omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & \sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix}\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = W\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

$$W^{-1} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \dfrac{\sin(\phi)}{\cos(\theta)} & \dfrac{\cos(\phi)}{\cos(\theta)} \end{bmatrix}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \dfrac{\sin(\phi)}{\cos(\theta)} & \dfrac{\cos(\phi)}{\cos(\theta)} \end{bmatrix}\begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

If the Euler angle is small enough (around 0 degrees), matrix W will be an identity matrix, thus the angular rate will equal to Euler angle rate.

The equation of movement can be defined as following:

$$m\ddot{X}^G = F_g - F_T^G - F_d$$

$$\ddot{X}^G = \begin{bmatrix} \ddot{X}^G \\ \ddot{Y}^G \\ \ddot{Z}^G \end{bmatrix} \; and \; F_g = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}$$

$$F_d = \begin{bmatrix} K_{dx} & 0 & 0 \\ 0 & K_{dy} & 0 \\ 0 & 0 & K_{dz} \end{bmatrix}\begin{bmatrix} \dot{X}^G \\ \dot{Y}^G \\ \dot{Z}^G \end{bmatrix}$$

$$F_T^G = R_b^G \, F_T^b$$

$$F_T^b = \sum_{i=1}^{4} F_i$$

$$F_T^G = R_b^G \sum_{i=1}^{4} F_i = \begin{bmatrix} 0 \\ 0 \\ K_T \sum_{i=1}^{4} \omega_i^2 \end{bmatrix}$$

The equation of rotation can be defined as following:

$$J_b \dot{\omega} = \tau_m - \tau_g - (\omega * J_b \omega)$$

$$J_b = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix} \dot{\omega} = \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} (\omega * J_b \omega) = \begin{matrix} \dot{\theta}\dot{\psi}(J_x - J_y) \\ \dot{\psi}\dot{\phi}(J_z - J_x) \\ \dot{\theta}\dot{\phi}(J_y - J_z) \end{matrix}$$

$$J_m \, \dot{\omega} = \tau_m - \tau_\psi$$

$$At\ hover\ state, \dot{\omega} = 0\ so\ \tau_\psi = \tau_D$$

$$\tau_D = \left(\frac{1}{2}\right) R \rho C_D A (\omega R)^2 = K_d \omega^2$$

$$\tau_\psi = \tau_D = (-1)^{i+1} K_d \omega_i^2 = K_d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)$$

$$\tau_{\phi,\theta} = \sum r * T$$

$$\tau_{\phi} = lK_T(\omega_4^2 - \omega_2^2)$$

$$\tau_{\theta} = lK_T(\omega_1^2 - \omega_3^2)$$

$$\tau_m = \begin{bmatrix} \tau_{\phi} \\ \tau_{\theta} \\ \tau_{\psi} \end{bmatrix} = \begin{bmatrix} lK_T(\omega_4^2 - \omega_2^2) \\ lK_T(\omega_1^2 - \omega_3^2) \\ K_d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix}$$

$$\tau_g = \omega * G_z \sum_{i=1}^{4} J_{\tau}\omega_i$$

$$\tau_g = \begin{bmatrix} \dot{\theta} \\ -\dot{\phi} \\ 0 \end{bmatrix} J_{\tau} \sum_{i=1}^{4} (-1)^{i+1}\omega_i = \begin{bmatrix} J_{\tau}\dot{\theta}(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ -J_{\tau}\dot{\phi}(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ 0 \end{bmatrix}$$

ω is the angular velocity, $\dot{\omega}$ is the angular accelerations, J_b is the body moment of inertia, J_m is the motor moment of inertia, J_i is the moment of inertia in i axis, τ_m is the body torques from motors, τ_g is gyroscopic effect torques.

R is the radius of propeller, A is cross-section of propeller, ρ is air density and l is length of the quadcopter arms.

4 Basic algorithm for quadcopter system

We are running a real time operating system (Free RTOS) on the ESP32 and it will allow the system to run different function at the same time and doesn't interfere with each other.

We assigned the frequency of each task and the max running time, if the task got trapped in a task over the max running time, the processor will kill the task and go to the next one. And if the task doesn't as much time as defined by the frequency, the task will wait for it. And we will also make sure that the key functions will always have the highest priority. The frequency of each task is defined in Table 4.1:

Name of the task	Frequency of the Task
MPU	200Hz
Motors	400Hz
Wi-Fi	200Hz
Status_check	100Hz
Flight_Mode	10Hz

Table 4.1 Frequencies of tasks

The ESP32 has a dual core processor and the certain task in the RTOS can be assigned to different cores. We will assign the MPU task to core1 and assign MOTOR and Wi-Fi tasks to core 0.

As shown in Figure 4.1, the first task of the system will be the initialization, it will initiate the MPU6050, ESCs and Wi-Fi, a system error flag will be set if there is any part of the system that is not running, and it will send the error message through the Wi-Fi. After the initialization and the system check the quadcopter will be ready for the flight and sending the status to the ground station through the Wi-Fi.

The command from the ground station will determine the flight mode (Table 4.2) of quadcopter and the quadcopter will start.

Name of the flight mode	Function of the flight mode
Takeoff	Raise the quadcopter to a certain height, go to Hover mode automatically when reached the desired height
Hover	Hold the altitude in the hover state.
Landing	Set the desired altitude to 0, stop all motors when reached the ground.

Table 4.2 Flight Modes

MPU loop will always feed attitude data to the motor loop. The stabilize controller inside the motor loop will process the current data with the desired attitude to and generate the PWM signals to the ESC. At the meantime, the MPU loop will also send the flight data to the Wi-Fi loop to give the feedback to the ground station.

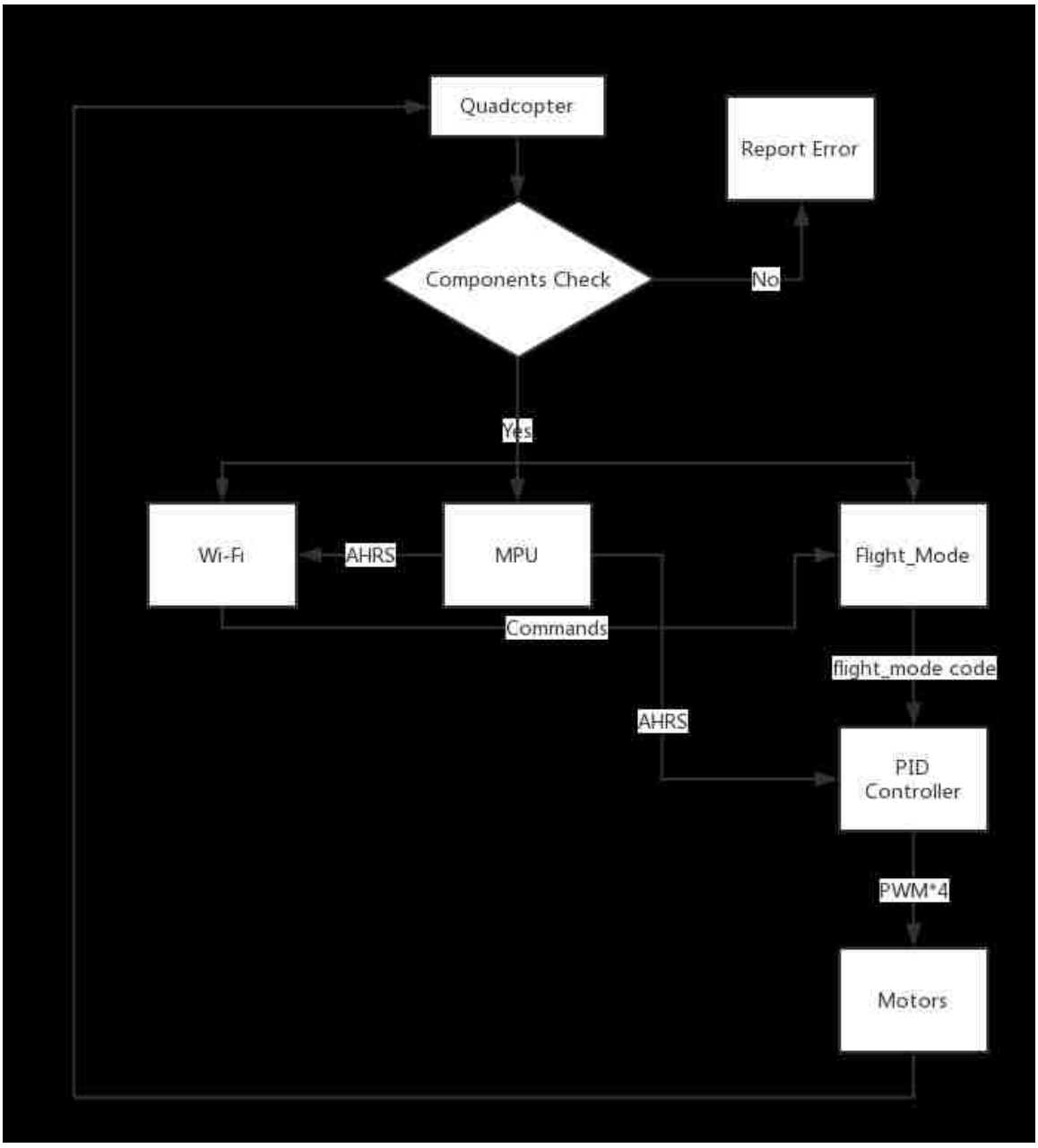


Figure 4.1 Basic algorithm of quadcopter

5 Tuning the PID controller for attitude

5.1 Basic idea

To control the attitude of the quadcopter, we will need to apply a PID controller to it. The first thing we are doing is using a single PID controller for Euler angle. As shown in Figure 5.1, input for the controller will be the error between the current attitude and the desired attitude. This error will be fed to the PID controller and the PWM for each motor will be the output. The IMU will gather the data of the quadcopter as a real-time feed back to the PID controller.

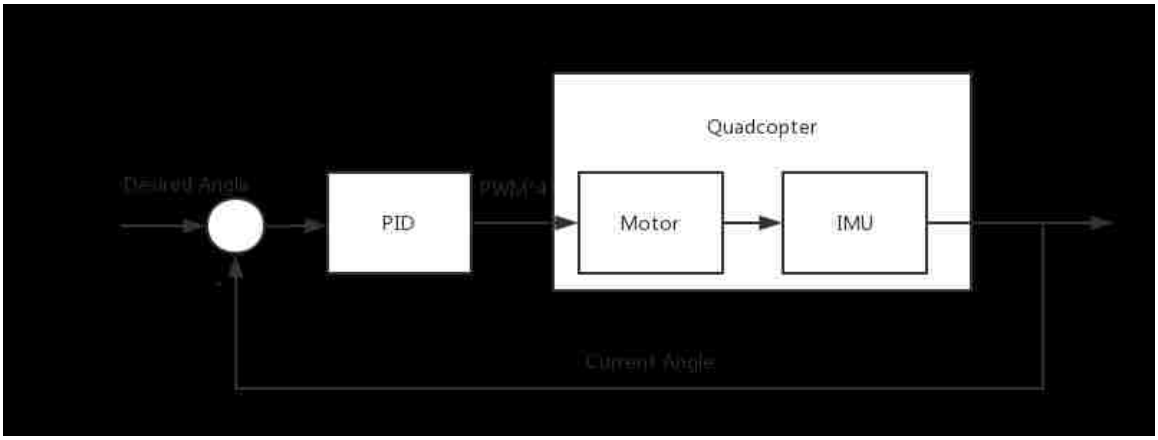


Figure 5.1 PID controller for angle

Only controlling the Euler angle is not enough, to add a damper to the system, we will apply a cascade PID controller. As shown in Figure 5.2, the outer loop is angle controller and the inner loop is the angular rate controller. The outer loop will have the error of desired Euler angle and the current Euler angle as the input, and the output will be the desired angular rate. The inner loop will take the error of current

angular rate and the output from the inner loop as the input of the inner loop. This controller has a better performance in the practice and it is easier for tuning.

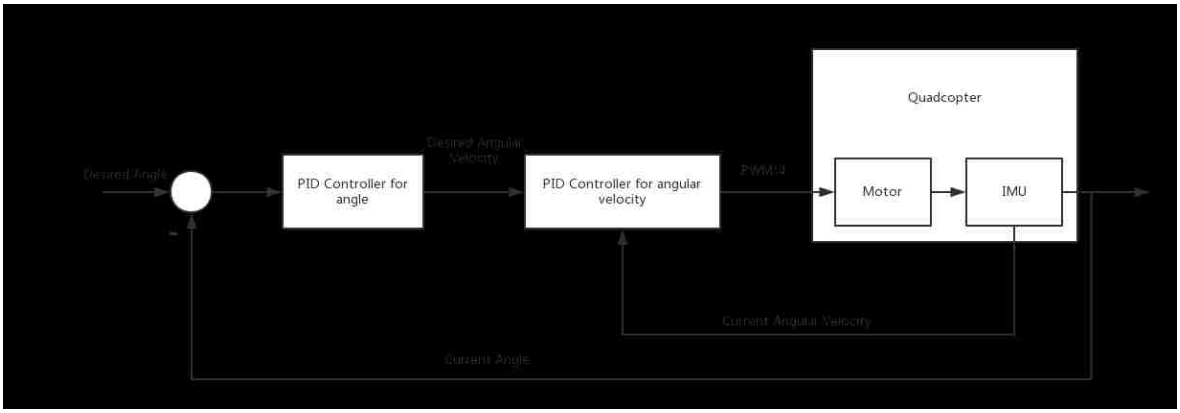


Figure 5.2 Cascade PID controller

5.2 Equipment of tuning the PID

Since the quadcopter can't perform a safe flight before the controller is well tuned, we must tune the controller before the flight. We started tuning controller on single axis with our tuning kit (Figure 5.3).

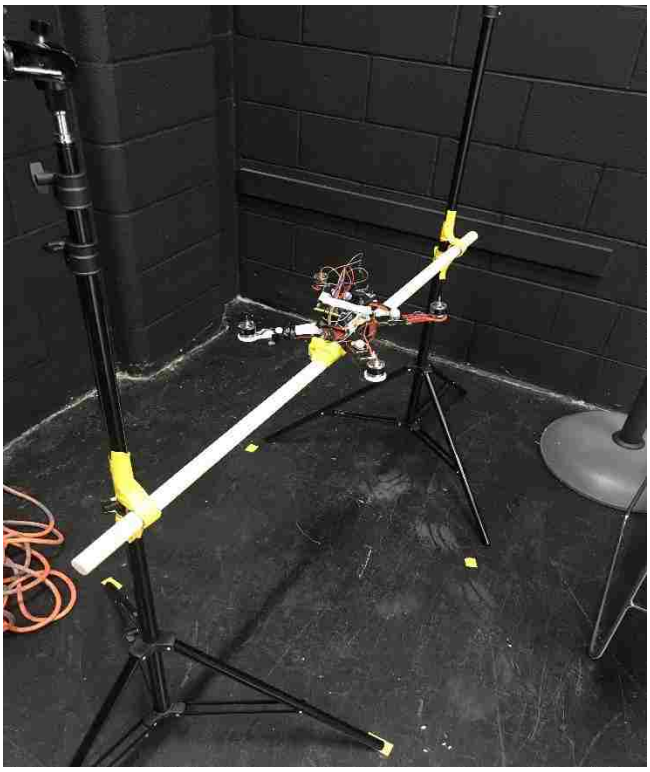


Figure 5.3 Tuning Kit

With this tuning kit, we can attach the quadcopter on the stick, and then put the stick inside two rings on the rack. The quadcopter will only have one degree of freedom in this situation and it is safe even the controller goes crazy. We started with tuning

5.3 Design of online tuning system

Using ESP32 as the flight controller also enabled the online tuning feature since we can have the entire control of the communication with ground station. We must set the PID coefficients before the flight in Pixhawk (Ardupilot), and those coefficients

are fixed during the flight. Each upload of the firmware will take at least 40 seconds to compile and upload. Since tuning the PID controller is a time-consuming task, the design of online tuning system will save remarkable time.

In the online tuning system, PID coefficients can be changed during the flight. The ground station will send all the coefficients to quadcopter at 100Hz and the quadcopter will update coefficients if there is any change.

5.4 Tuning the inner loop

The inner loop is controlling the angular velocity of the quadcopter. We can set P gain to a certain value, and because the PID coefficients are all zero for the outer loop, the desired angular velocity will be zero in the inner loop, the ideal result will be that the quadcopter will be stable at any angle. On the quadcopter testing rack, the motors will only provide torque on a certain axis, thus, the quadcopter should balance itself at any angle.

5.4.1 P_{inner}

We changed the P_{inner} from 0, with the P_{inner} goes higher, it is easy to observe that it's getting harder to change the roll angle of the quadcopter. If P_{inner} is too much, we can observe the overshoot of the inner loop, there will be oscillation if a

disturb is applied. The quadcopter will slightly turn around the stick because of the existence of steady-state error of inner loop. In that way the angular velocity will not be zero and quadcopter will turn around.

5.4.2 I_{inner}

Then we need to add I_{inner} to compensate the stead-state error of inner loop. With ranging from 0, quadcopter will gradually hold the position. If the I_{inner} is too much, the rotation of the quadcopter will diverge under a small disturb. With $P_{inner} = 0.12, I_{inner} = 0.03$, the quadcopter could stable itself at any angle on the rack.

5.4.3 D_{inner}

The D_{inner} comes from the difference between the difference of this error and last error. The derivative of angular velocity is angular acceleration, the vibration of the quadcopter itself if very strong and this will lead to the a very high noise to the gyro readings and the angular acceleration will be remarkable. We can apply some filter to the data. The increase of D_{inner} will not change the dynamic of the quadcopter too much, it's easier to observe that it's getting smoother when going towards the target. The D part of the inner loop is just a supplemental thing, and if the vibration of the frame is too much, we can cancel the D_{inner}.

5.5 Tuning the outer loop

5.5.1 P_{outer}

Increase the P_{outer} from zero and we can observe that quadcopter is going back to the middle. If we put the quadcopter to some angle, it will start to go back. The response time will get shorter if P_{outer} goes higher. If P_{inner} is too much, the oscillation will start to diverge.

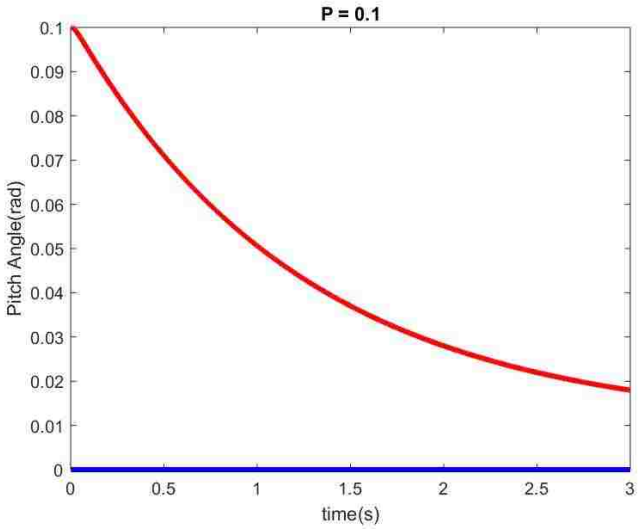


Figure 5.4 Output of Pitch angle when P=0.1

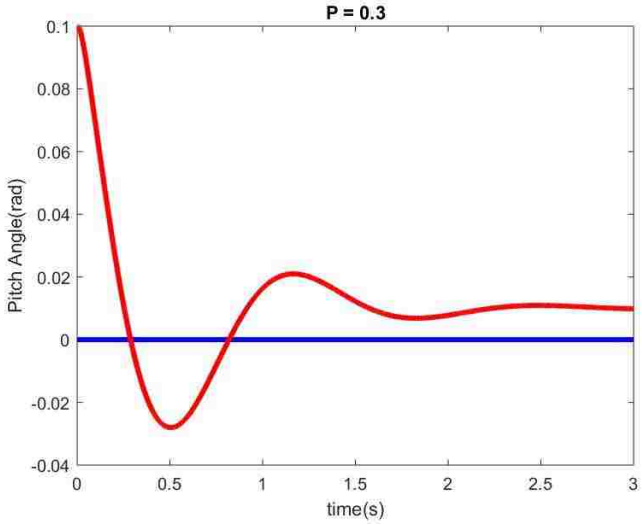


Figure 5.5 Output of Pitch angle with P=0.3

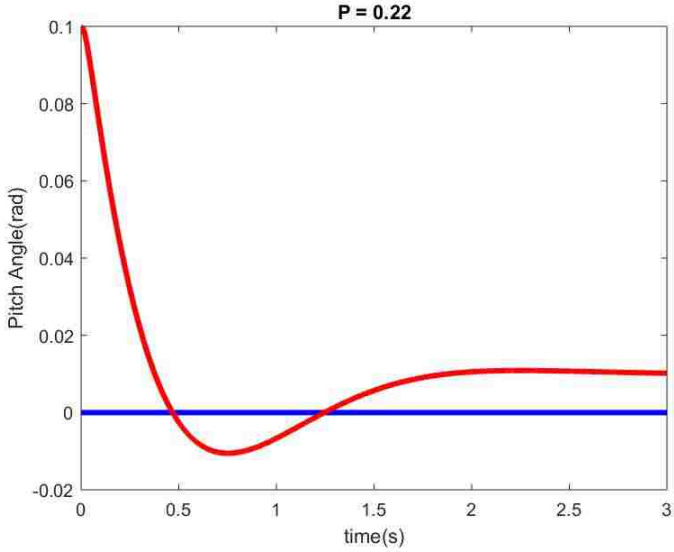


Figure 5.6 Output of Pitch angle with P=0.22

5.5.2 I_{outer}

I_{outer} will eliminate the steady-state error of the equilibrium state of the quadcopter. And the oscillation will diverge if the I_{outer} is too much. As shown in Figure 5.7, the controller will compensate the steady-state error when $I_{outer} = 0.02$ is applied.

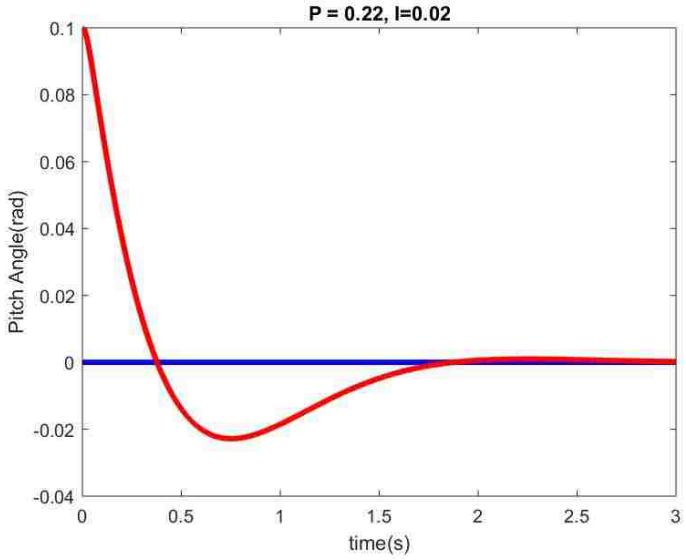


Figure 5.7 Output of Pitch angle with P=0.22, I=0.02

5.5.3 D_{outer}

As shown in Figure 5.8, the controller will have a shorter response time and less overshoot if D_{outer} is applied.

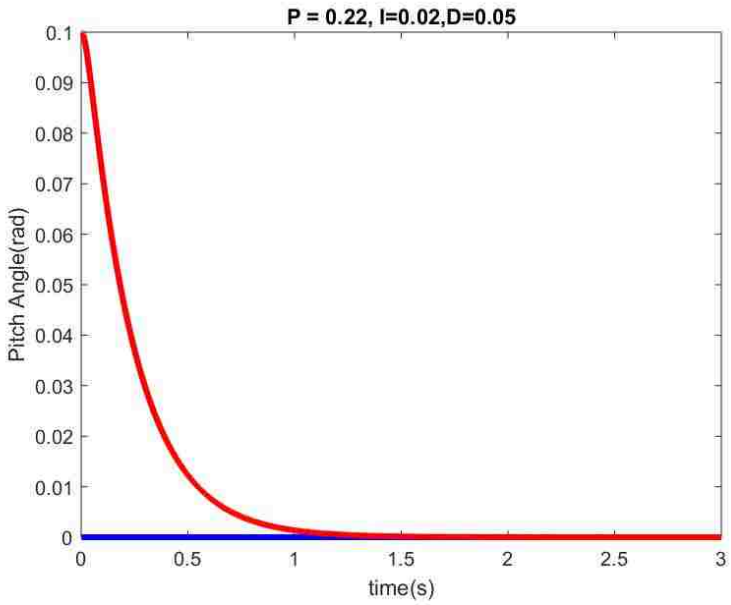


Figure 5.8 The output of Pitch with well-tuned PID

6 Tuning the PID controller for altitude

6.1 Using barometer

The altitude of the quadcopter can be measured by the barometer attached with the GY-87 board. It's a Bosch Sensortec BMP 180.

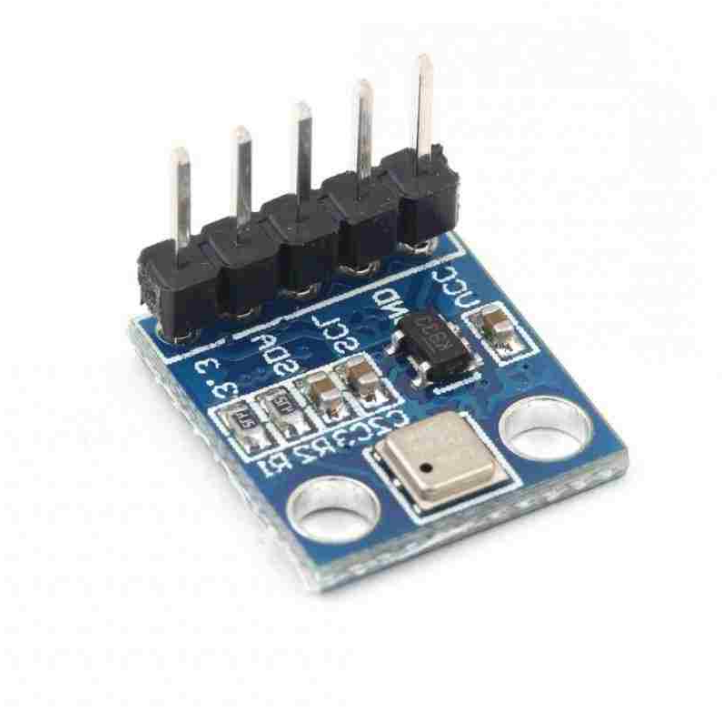


Figure 6.1Bosch Sensortec BMP 180

The barometer is measuring the altitude by comparing the air pressure of the current and compare with the local standard air pressure to calculate the altitude. This works fine in outside, but it will be affect a lot by the temperature and the weather.

The reading indoor is too inaccurate because the temperature and the moisture is changing a lot. There is an error about 30cm when stationed and the reading is drifting all the time.

6.2 Using sonar

The other approach is using the sonar HC-SR04, and Milad Habibi helped with most of the work of sonar. The sonar is working by emitting sound pulses and detecting or measuring their return after being reflected.

Figure 6.2 HC-SR04

This economical sensor provides 2cm to 400cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm. The detecting range is good enough for the indoor flight. Thus, we can use the sonar for the altitude measurement.

6.3 Tuning PID

6.3.1 P

Increase P_a from 0, by holding it with hands first, only power up the flight controller and the motor will not spin. Set the target altitude as 20 cm above the ground for safety. We can see that PWM outputs from flight controller is changing with height.

Set the target as 1 meter above the ground, with the increase of the P_a. If P_a is too small, the quadcopter will never reach the 1-meter height, and if the P_a is too large, the quadcopter is oscillating around the 1-meter level.

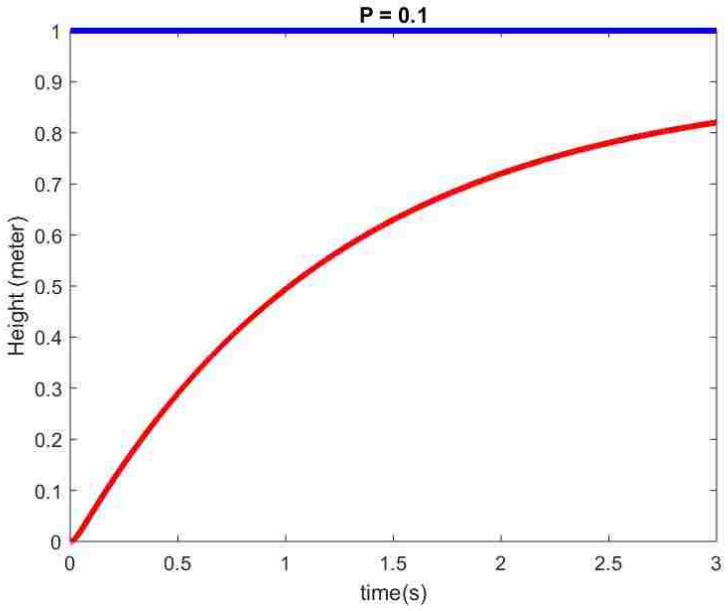


Figure 6.3 Output of height with P=0.1

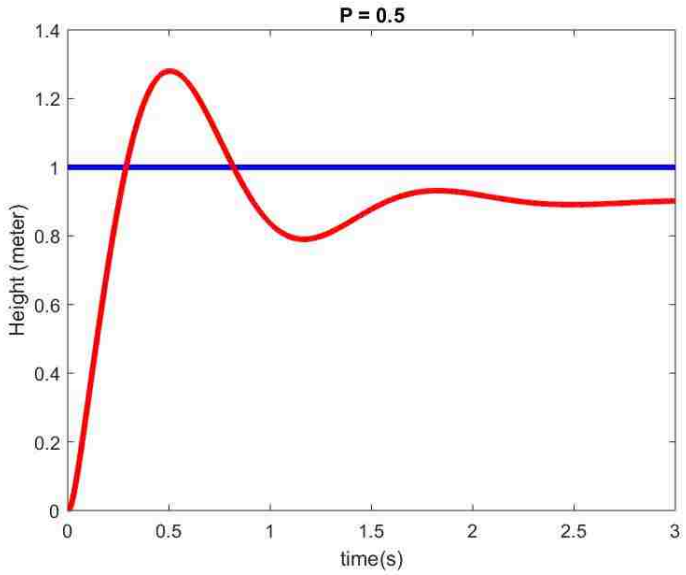


Figure 6.4 Output of height with P=0.5

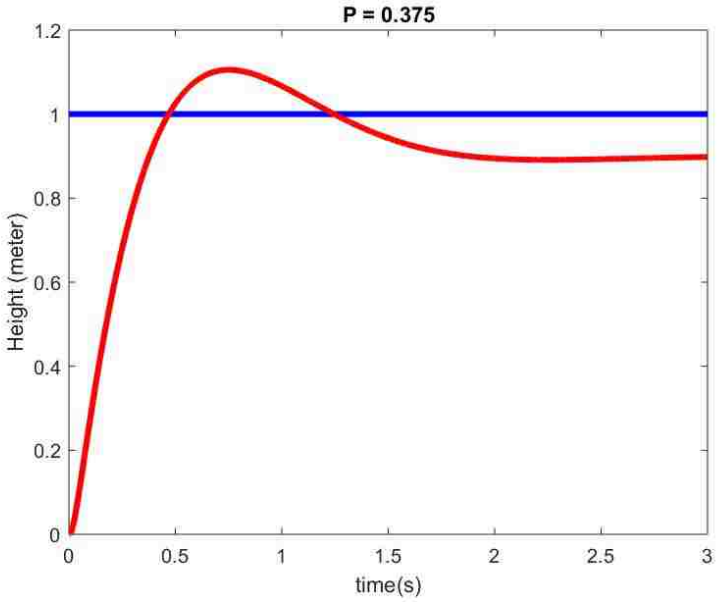


Figure 6.5 Output of height with P=0.375

6.3.2 I

Compare Figure 6.6 with Figure 6.5, quadcopter is not exact 1 meter high, even the P_a is good enough, there are steady-state errors of the altitude controller. With the increase of I_a the quadcopter will be gently ended up at reaching the height of 1 meter. If the I_a goes higher, the controller will diverge.

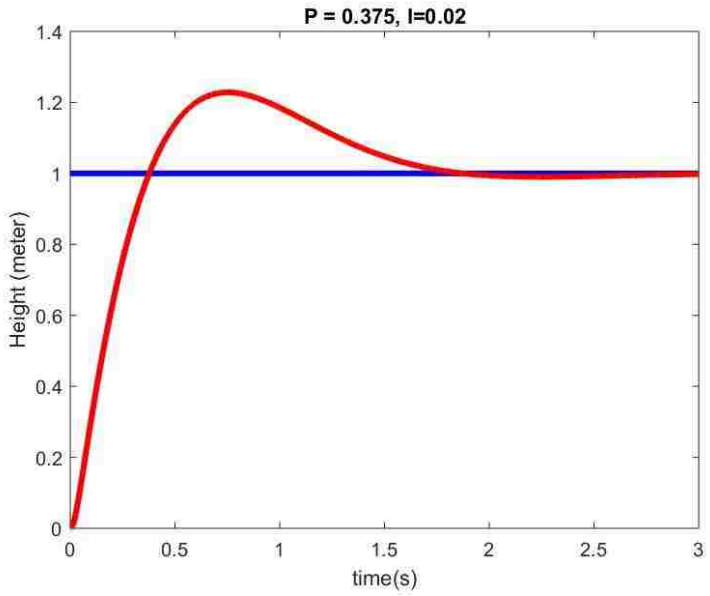


Figure 6.6 Output of height after I_a is applied

6.3.3 D

By increasing the value of D_a , the approaching motion will be smoother. And as shown in Figure 6.7, it compensates out the overshoot and shorten the response time.

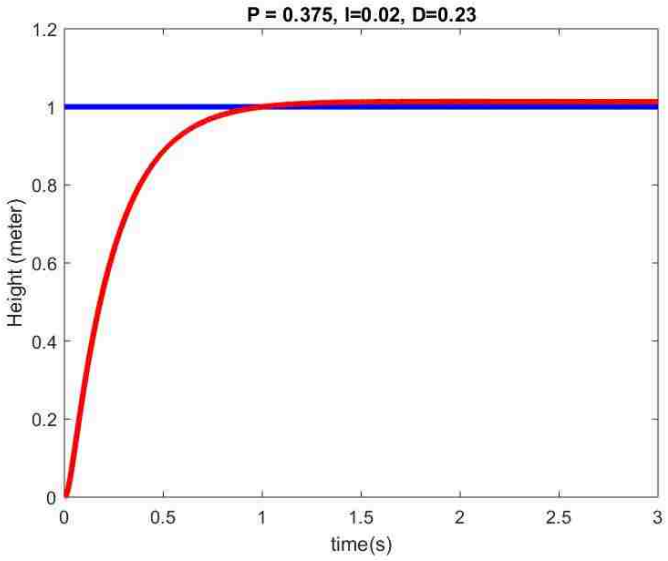


Figure 6.7 Output of height of a well-tuned PID controller

7 Test Flight

After tuning all the controllers onboard, we started several test flights, about 50% of the test didn't start because some hardware is not initialized when powered up. And about 20% of the flight tests crashed for unstable heading. About 30% of the test went well and finished the following task:

a) Takeoff to the height of 1 meter;
b) Hover until 13 seconds after takeoff;
c) Land to the ground;

The desired pitch and roll angle and angular velocity are all set to 0, the quadcopter should perform a straight takeoff, hover in the certain level and then land to the ground.

As shown in Figure 7.1, the altitude controller has some overshoot when taking off, this is due to the battery voltage level is decreasing, the gain of the controller should be changing with the voltage level instead of being fixed. The response time is about 1.7 seconds, this problem can be fixed in the future if add a battery monitor to the quadcopter to measure the voltage level.

The pitch and roll angle are oscillating around 0.01 rad in the real flight as shown in Figure 7.2 and Figure 7.3, this might due to the vibration of motors and the low frequency of the MPU loop. We could reach a 400 Hz MPU sampling frequency

if use Motion Capture system or use our own filter instead of using DMP. Both of them will increase the quality of the controller with a high sampling rate.

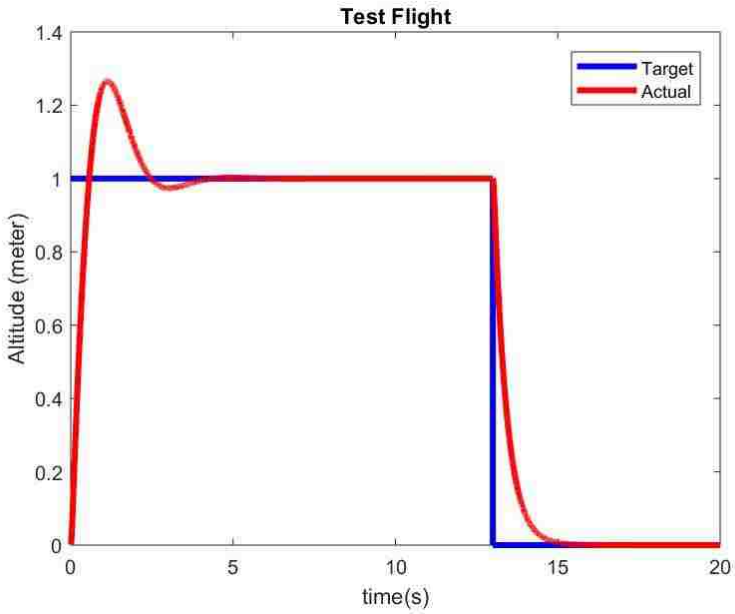


Figure 7.1 Altitude test result

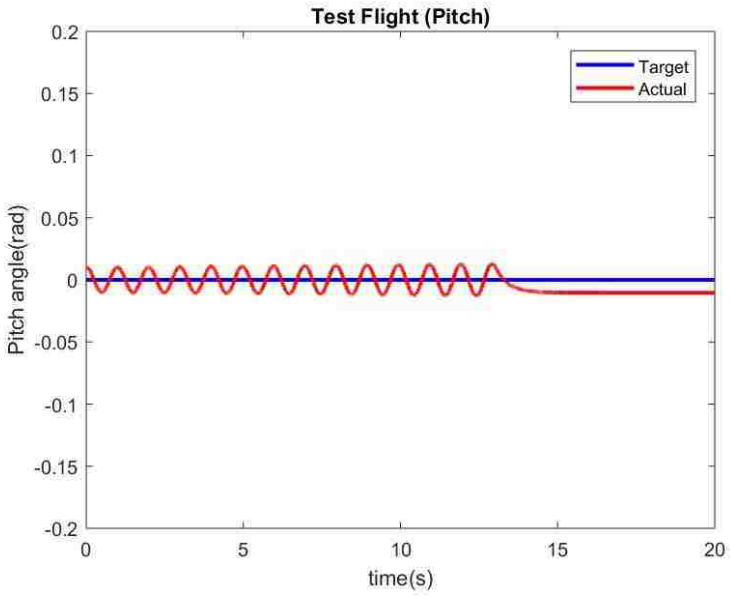


Figure 7.2 Pitch angle test result

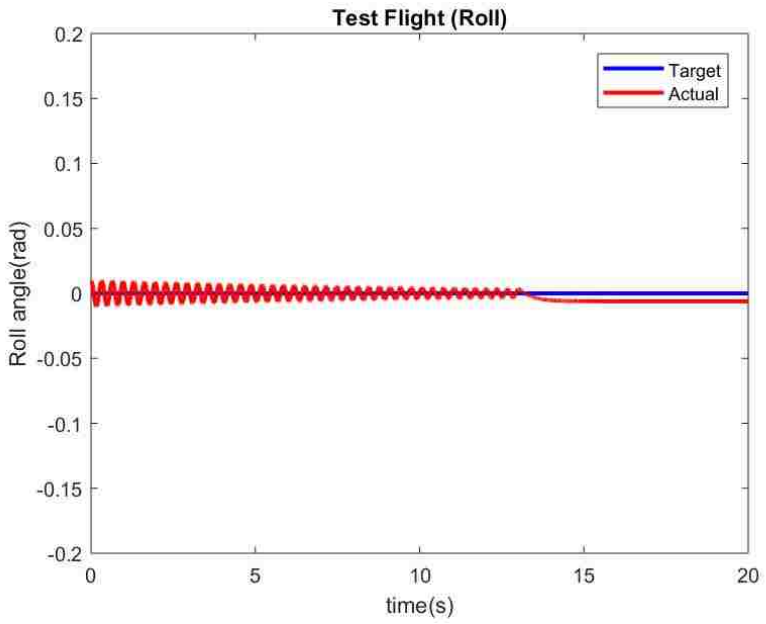


Figure 7.3 Roll angle test result

There is another problem during the test flight is that the yaw angle is not fixed. The quadcopter will perform yaw movement during the hovering. Since the indoor magnetic field is complex and variant, the heading reading form HMC5883L is not accurate enough. And this is the main reason for the crashes during the test.

8 Conclusion

Our new quadcopter was manufactured and tested. It has the cheapest price in the range of research use and variety of potential modification in the future. Taking-off, hovering and landing are performed very well in several tests.

There are still some problems to be solved, the reliability of hardware is still too weak for the regular research use and the reading of yaw angle is not stable for the indoor flight since the magnetometer is not accurate inside.

We are now working on enabling Motion Capture system with this new quadcopter through Wi-Fi communication, this feature will fix the yaw reading of the quadcopter and provide attitude information at a higher sampling rate which will lead to a better flight performance of the new quadcopter.

Appendix A

Specifications of ESP32

Categories	Items	Specifications
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps)
		A-MPDU and A-MSDU aggregation and 0.4 µs guard interval support
	Frequency range	2.4 GHz ~ 2.5 GHz
Hardware	Operating voltage/Power supply	2.7 ~ 3.6V
	Operating current	Average: 80 mA
	Operating temperature range	-40°C ~ +85°C
	Package size	18±0.2 mm x 25.5±0.2 mm x 3.1±0.15 mm

Appendix B

Specifications of F330 quadcopter frame

Model	F330
Frame arm size (mm)	150x20x30
Diagonal Wheelbase (Motor to Motor) (mm)	330
Total weight (g)	156.00

Appendix C

Specifications of Motors

SKU	L2210A-1650	Brand	Turnigy
Shipping Weight(g)	92.00	Length(mm)	95.00
Height(mm)	55.00	Kv(rpm/V)	1650
Max Currents (A)	17.50	Power (W)	180.00
Length B (mm)	25.00	Can Length D(mm)	12.00
Max Voltage(V)	11.00	Shaft A(mm)	3.00
Unit Weight (g)	49	Diameter C(mm)	28.00
Total Length E(mm)	44.00	Resistance (mΩ))	0.00

Appendix D

Specifications of ESC

Constant Current (A)	20A
Input Voltage (V)	12 (2-4 cell Lipos)
BEC	OPTO
MCU	Arm Cortex-M0
Weight (g)	6
PCB Size (mm)	22 x 12

9 Biography

Guangyi Liu

Education

Department of Mechanical Engineering, Lehigh University

M.S., Mechanical Engineering **09.2016-06.2018**

School of Aerospace Engineering, Beijing Institute of Technology

B.E., Aerocraft Design and Engineering **09.2012-06.2016**

Exchange Student, Technische Universitaet München **02.2016-06.2016**

Research& Project Experiences

✓ **Indoor UAV research in Lehigh Distributed Control and Dynamical Systems Laboratory**

Research Assistant, Advisor Prof: Nader Motee and Babak Shirmohammadi **02.2017-Now**

This project builds our DIY quadcopters with Motion capture system to accomplish the indoor and high accuracy flight.

- Fixed the onboard sensor drift caused by complex indoor magnetic field.
- Built our own flight controller after studying and modifying Arducopter firmware.
- Took part in building communication between ground station and quadcopters through Wifi modules.
- Took part in building Motion Capture system in the lab and hardware communication debugging.

✓ **Bachelor Thesis done at Technische Universitaet München**

Advisor Prof: Mirko Hornung and Gilbert Tay **02.2016-06.2016**

Title: *Network Analysis and Geographical Evaluation of Air Transportation Systems and Pre-Defined Aircraft Clusters*

- Analyzed the robustness of air transportation system in China, Europe, South East Asia and United States and the reason of the system failure.
- Analyzed the performance and efficiency on each airport in the region included above and the network features of different pre-defined aircraft clusters.

✓ **Design of Fireworks System (Funded by the Chinese Outstanding Engineers Training Program)**

Research Assistant, Advisor Prof: Wen li **07.2015-09.2015**

This project is to design the automatic launch system and conduct dynamic simulation of the whole process including emission, ignition, releasing the parachute, and landing.

- Took part in the writing of classes including igniter and the simulation of flight

environment.

- Completed the design of whole structure, writing of general program code, skillfully grasped the numerical computation solving differential equations and interpolation fitting.

✓ **Course Project of Flight Dynamics** ***Advisor Prof: Hai Lin***

Lateral Stability Analysis of Aircraft **05.2015-08.2015**

- Studied coupled mode of Dutch roll and coupled modal space control for the coupling characteristics of transverse side direction movement.
- Built the five-freedom coupling dynamic model and analyzed the principle of generation of aerodynamic coupling, control coupling, kinematic coupling and inertia coupling.
- Established the small perturbation model of aircraft with linear method, further analyzed the influence factor of control derivative.

Design of Guiding Laws **04.2015-05.2015**

- Studied the principle, composition of guidance control system, investigated the proportional navigation and multi-model composite homing guidance.
- Designed the algorithm with C based on the composite mode of radar and infrared seeker.

Internship

Taiyuan Mining Machinery Group CO. Ltd. **06.2015-09.2015**

Assistant Engineer *Mentor: Jianhua Su*

- Studied the structure and working principle of crawler travel mechanism, and conducted the theoretical calculation of running resistance to determine the external load for dynamics simulation.
- Investigated the reliability failure analysis, and found out the main failure points of continual coal winning machine.
- Explored to build the virtual prototype model based on the dynamic theory of multibody system.

Technical Skills

- Programming Languages: C, C++
- Proficient in Matlab, AutoCAD, Gephi and embedded programming with Arduino