8-2017

Atmospheric Microbial Community Sampling System for Varying Altitude Collection

Kenneth David Domingue
Western Michigan University, kenneth.domingue@gmail.com

ATMOSPHERIC MICROBIAL COMMUNITY SAMPLING SYSTEM FOR VARYING ALTITUDE COLLECTION

by

Kenneth David Domingue

A thesis submitted to the Graduate College
in partial fulfillment of the requirements
for the degree of Master of Science in Engineering
Mechanical and Aerospace Engineering
Western Michigan University
August 2017

Thesis Committee:

Kristina Lemmer, Ph.D., Chair
Kathryn Docherty, Ph.D.
Peter Gustafson, Ph.D.

ATMOSPHERIC MICROBIAL COMMUNITY SAMPLING SYSTEM FOR VARYING ALTITUDE COLLECTION

Kenneth David Domingue, M.S.E.

Western Michigan University, 2017

The study of airborne microbial communities in the vertical atmosphere is an area of research in which very little data has been collected and analyzed. Developing a vehicle to house biological sampling equipment that collects, and mitigates contamination is required by a team of biology researchers to perform sampling of atmospheric microbial communities. This study uses a sampler box that is attached to aerostatic vehicles: zero pressure balloons and tethered balloon/kite structures. Zero pressure balloons are used for sampling at altitudes above 150 m above sea level, and tethered balloon/kite structures are used for sampling at 30 m and 150 m above ground. Sampling also includes a ground-based platform at 2 m above ground. The sampler box contains petri dishes that are used to collect atmospheric microbial biomass from which community DNA is extracted. Material selection requirements were set by the biologists to ensure the samples are not contaminated by the sampler box and were reusable. The sampler box was designed and built to follow Federal Aviation Administration regulations on unmanned free and moored balloons. Therefore, the sampling box weighs less than 6 pounds. HAM radios are used to communicate with and track the sampler box. Accounting for all of this, a sampling system for varying altitude collection was developed.

ACKNOWLEDGEMENTS

The work of this thesis has been done with numerous people and in this section, I want to extend my utmost thank you for the contributions, support, and help.

I want to start with my parents, David and Linda Domingue, for you guys have always been there for me. The both of you have helped me in countless ways and I for that, I am extremely grateful. My wife, Virginia Domingue, you have been my biggest supporter outside of my research team. You have listened to all the problems and solutions to every aspect of this thesis. There numerous times that you have helped me out with along this adventure, and I cannot thank you enough for that. To my family and my wife, thank you and I love all of you.

To Dr. Gustafson for being a part of my committee and always having his door open to talk to. From the several classes that I have taken with you, I have learned a lot about how to structures and materials, but I think most importantly is you have taught me there is no absolutes with engineering and to defend and own your work.

There are 3 undergraduates that have worked closely with me on this research. Margaret Mooney, Thomas Kerber, and Allison Spring, have all been some of the hardest working people that I have ever known and have contributed to

this thesis more than I can describe in such a short section of my paper. Margaret and Thomas are both undergraduate aerospace engineers working on this project and without the hard work, determination, and true grit of these individuals, there is no way this thesis could ever be accomplished. Allison Spring was an undergraduate biology research (now in her master's program), and has spent long days sampling on roof tops, in tick invested fields, the frigid cold of Michigan winters and so on, while right after doing all of that, going into the biology lab and performing a DNA extract that lasts for full 8-hour days at times. She has been additionally one of the hardest working people I know and has been able to contribute to this thesis in numerous ways.

Dr. Kathryn Docherty, a member of my thesis committee, co-principal investigator of this research, and most importantly a true mentor on so many levels. Kathryn has the unimaginable determination to complete research and ability to drive creative ideas to solve real problems. Thank you for being a mentor and fellow researcher to me, and from you I have learned a lot about determination, teamwork, leadership, communication, and always estimate double the amount of time I think it will take to complete a project. There is numerous things to thank you for but the number one this is that you have made me a better engineer from the demands that you expected of this research, and for that I am extremely grateful.

Finally, Dr. Kristina Lemmer, my faculty mentor for my thesis, co-principal investigator of this research, leader of team engineering, and most importantly the best mentor I could have. You have pushed me to work the hardest I have ever worked, to understand and give logical reasoning behind my work, and challenge myself to come up with creative solutions to solve problems. You have worked with me since the start of this thesis and we both have had our victories and defeats along the way. There is with no doubt that because of you, I am a better engineer, researcher, and person. This thesis has been challenging but will be something that I will never forget. Thank you for giving me this opportunity and working with me on this adventure. Thank you!

Kenneth David Domingue

TABLE OF CONTENTS

CHAPTER

CHAPTER

LIST OF TABLES

LIST OF FIGURES

List of Figures—Continued

LIST OF ACRONYMS

AMC	Airborne Microbial Community
APRS	Automatic Packet Reporting System
DTMF	Dual Tone Multi-Frequency
CAD	Computer Aided Drawing
FAA	Federal Aviation Administration
GPS	Global Positioning System
RC	Remote Control
SP	Super Pressure
ZP	Zero Pressure

CHAPTER I

INTRODUCTION

Soil and water microbial communities are often-tested biological habitats where DNA can easily be extracted from soil and water samples collected from desired locations. Researchers use data from these samples to understand abiotic and biotic interactions between microbial communities and the environment. The atmosphere is another potential microbial habitat that should be tested because of the potential for local and global distribution of microorganisms through the air. However, very little research has been performed to investigate airborne microbial communities (AMCs) to determine what communities exist in the atmosphere. The goal of this work was to design, build and test a sampling mechanism to aid in the discovery of relationships between ecosystem, altitude, season, atmospheric layer, and microbial communities.

A collaborative research team of biologists and engineers was created to develop a sampler system that collects AMCs. The biology team members set requirements and criteria for the sampler system. These requirements include material limitations, re-usability, and prevention of sample contamination. The engineering team members used the requirements, combined with Federal Aviation Administration (FAA) regulations and safety restrictions to design and develop a method for obtaining the samples using balloon borne sampling boxes.

This chapter begins with a brief discussion of the project scope. Next is a history of ballooning, followed by a description of buoyancy and the different types of modern scientific balloons. Finally, there is a discussion of related experiments.

Following this introductory chapter, the sampler boxes are described. In chapters 3 and 4, the tethered and launched balloons, respectively, are discussed. Chapter 5 gives details of the experiments that were performed to verify functionality of the sampling boxes, and chapter 6 gives results of those functionality experiments. Finally, chapter 7 provides conclusions and suggestions for future work.

Scope

The scope of the project was to design a system capable of sampling airborne microbes from within a specific altitudinal range, without introducing human or ground-level contamination of the AMCs. The sampler box is the primary vessel for transporting the microbial samples from sampling altitudes to the biologists. Several criteria must be met to ensure the sampler box meets project requirements and stays within FAA compliance. The developed tethered and launched sampling systems carry the sampler boxes to their respective altitudes, and maintained there. Additional subsystems for the launched and tethered systems that allow for automatic opening and closing of the sampler box, tracking the sampler box, and overall safety were developed. Finally, the sampler box was developed to minimize the risk of contamination while providing a cost-effective solution for the materials and processes required to make the sampling system.

History of Ballooning

The first test of a large scale balloon was an unmanned, hot air balloon flown in June 1783 [1]. Flown in Annonay, France, the balloon was 700 m^3, and achieved a maximum altitude of approximately 2,000 m. The inventors of this balloon were J.M. Montgolfier and J.E. Montgolfier (brothers). They furthered their experiments and would inspire the work of César Charles to develop a hydrogen sealed balloon in August of 1783, Figure 1. This balloon was sealed at the bottom, and as a result, the increased pressure from the expanding hydrogen inside of the balloon during ascent caused the membrane to burst. Following this flight, a venting system was used to prevent bursting and later allowed for humans to fly on the balloons.

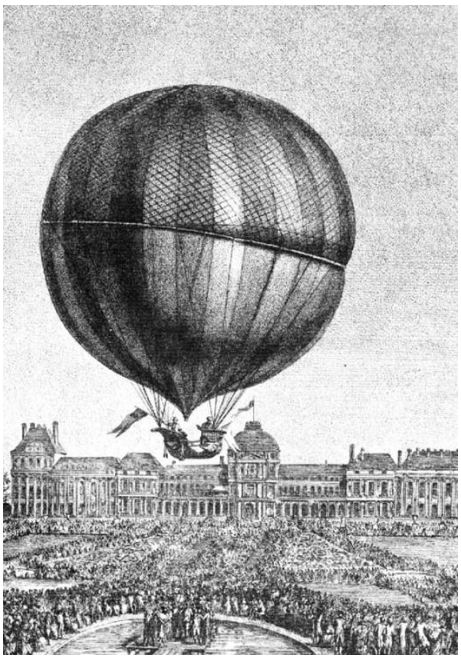

Figure 1: Hydrogen Balloon Developed by Charles in 1783 [1]

These hydrogen balloons were used by explorers and scientists in the early 1800s for scientific observations, such as atmospheric changes with altitude [1]. The balloons could reach altitudes of 10 km.

Throughout the 1800s and early 1900s, these types of balloons were used for exploration and research. Modern scientific balloons were developed in the 1930s when low-density polyethylene was invented. Previously, the balloons were made from silk or neoprene impregnated fabrics that were heavy and not as strong as polyethylene [2]. Low-density polyethylene is thin, lightweight, and strong, and it is still being used for scientific ballooning today [1].

Cylindrical balloons are relatively easy to make. Construction of a cylindrical balloon requires two sheets of plastic film with the 4 edges fused together or a tube of plastic film with the ends fused. During tests between 1945 and 1951, it was found that the cylindrical balloon would not expand to its full volume near the base as payload weight increased. This suggested that the circumferential stress was zero in these sections of the balloons. To optimize these cylindrical balloons, the sections that did not fill with lifting gas were removed. In further efforts to optimize zero pressure balloons, tetrahedral balloons, as shown in Figure 2 were made based on the ease of manufacturing. For testing purposes, these balloons were used to investigate stresses on the balloon that would occur at the ceiling of flight. As tetrahedral balloons were further optimized to suspend heavier payloads and achieve higher altitudes, the “natural shape” balloon was being developed.

Figure 2: Tetrahedral Balloons [1]

In the 1950s, a researcher from the University of Minnesota developed equations for the natural shape of zero pressure balloons that are used to describe the ideal shape and stress experienced in the balloon material [3]. Previously, the shape of a balloon was developed based on a sphere atop a cone. The equations for zero pressure balloon shape primarily related the stress experienced by the material to the structural limitations the balloon could withstand. The "natural-shape" balloons, shown in Figure 3, have extra material so that when the balloon is fully expanded at its ceiling, the circumferential stress "or hoop stress" will only be experienced at the balloon's maximum diameter, or equator [2].

At sea level, helium in the balloon is not contaminated by external atmosphere readily. This is because the difference in density between helium and air at standard temperature and pressure is greater than that at higher altitudes. Furthermore, when the balloon is only partially filled, the bottom of it is collapsed

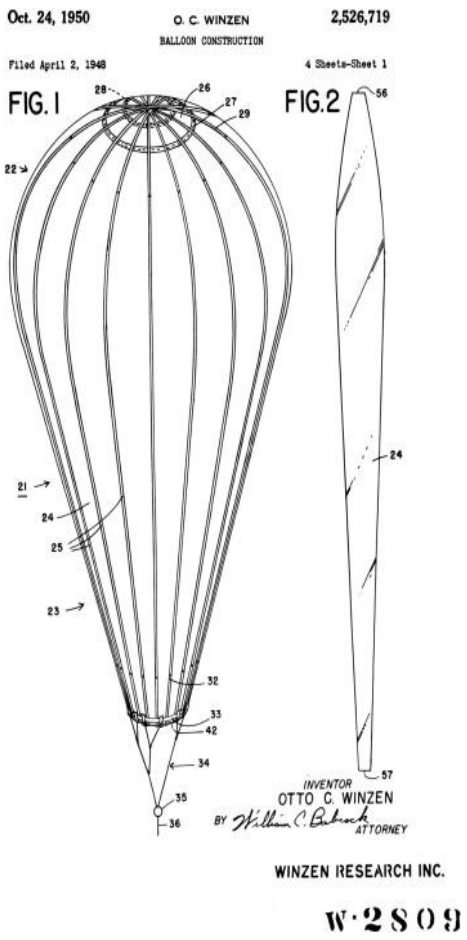

Figure 3: Natural Shape Zero-Pressure Balloon [16]

so that air cannot easily enter the envelop. As a zero pressure balloon ascends to higher altitude, air can contaminate the helium within a few hours, causing balloon performance to suffer, resulting in a loss of altitude. Ducts were implemented on the sides of the balloons in the gore sections that were relatively short, in comparison to the gore length, as illustrated in Figure 4. Ducts allow the bottom of the balloon to be sealed while still allowing lifting gas to vent. When the balloon reaches maximum altitude, the pressure inside of the balloon causes lifting gas to vent out the ducts. The pressure of the venting gas will be greater than that of the atmosphere on the outside of the balloon, preventing air from flowing into the

balloon [2]. When the pressure of the outside atmosphere and the lifting gas equalize, the ducts collapse, preventing air from entering the balloon. These ducts are typically placed near, and extend past, the bottom of the balloon [1].

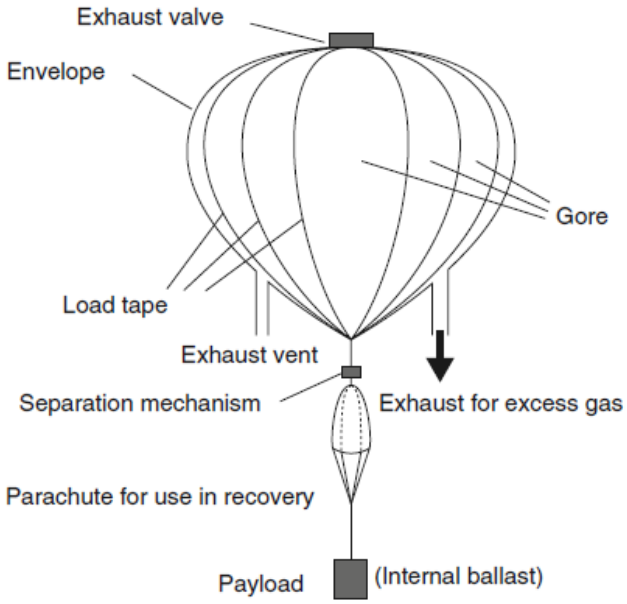

Figure 4: Zero Pressure Balloon Layout [1]

Until 1967, zero pressure balloons were only capable of floating for a maximum of 2 days. At that time, the use of ballast systems began to be employed to extend the duration of flights. Ballast systems unload approximately 7% of the mass of the combined payload and balloon every night to compensate for loss of lift during the night to day transition [4]. The dropped mass is typically iron particles or sand. To prevent having to use a significant amount of ballast, NASA began flying zero pressure balloons in Antarctica during the summer months of December 1991, so the balloon would not experience a night time transition. This extended flights to 42 days; however, the same balloon system would be far less functional if

flown closer to the equator [5]. Figure 5 shows a composition of the flight paths of the scientific balloon tests that NASA performed in Antarctica. There are 22 different flights shown in the image that occurred from December 1991 through December 2001. The longest flight lasted 42 days at a sustained altitude of 38 km [6].

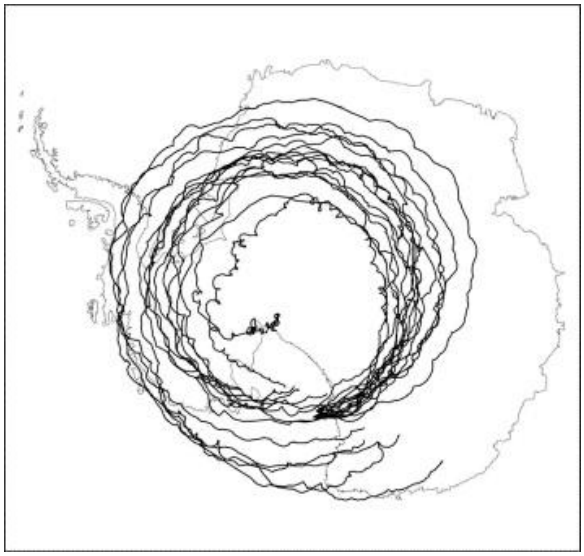

Figure 5: Flight Paths of all NASA Balloon Experiments in Antarctica [17]

While long distance zero pressure balloons were being tested in Antarctica, efforts to develop a super pressure balloon were in the works by NASA as part of their Ultra-Long Duration Balloon (ULDB) project [5]. Super pressure balloons are fixed volume balloons that are sealed from the atmosphere. These balloons are like the balloons that César Charles developed back in the 1700s, but with new material and technology advancements in ballooning. This prevents lifting gas from escaping from the balloon, and the pressure in the balloon increases as it ascends. As a result, the membrane and stress transferring features of a super pressure balloon

must be stronger than a zero pressure balloon to prevent bursting or leaking. Modern super pressure balloons are used for several different applications such as instrument testing and conducting research [7]. These balloons have the capability of reaching 33.5 km, and can fly for over 100 days while carrying over 2000 kg of payload. Figure 6 is a fully inflated super pressure balloon from NASA.

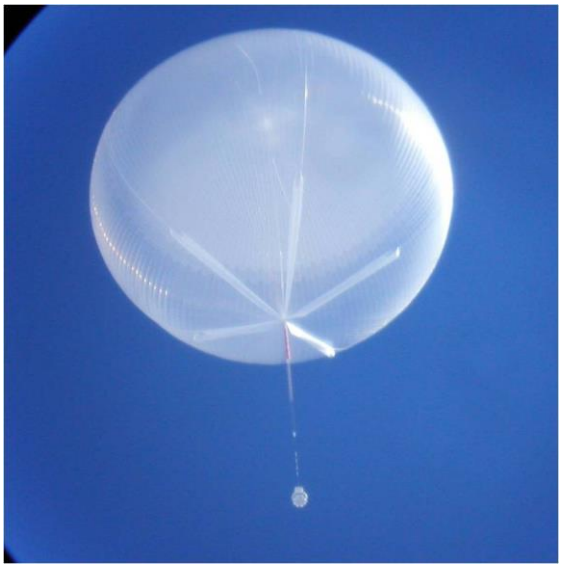

Figure 6: Fully Inflated Super Pressure Balloon [18]

With further advances in computational power, the ability to run simulations of stresses on a balloon has allowed researchers to design and develop new optimized balloons. These simulations include thermal performance [8], 3D simulations of the ascent and floating performance [9], and trajectory simulations [10]. As a result, modern scientific balloons have been able to carry heavier payloads, sustain altitude longer, and further the research of balloon performance.

Future ballooning research is focusing on research balloons for other planets. Researchers from the Jet Propulsion Laboratory, Wallops Flight Facility, Near

Space Inc., and Raven Industries have been working on developing a ballooning system for Mars [11]. The design allows scientific instruments, such as high resolution imaging, magnetic field mapping and sub-surface radar mapping, near the surface of Mars to cover long distances. Obtaining these data on Mars is beneficial for future rover and manned Mars missions.

Buoyancy

Balloons are built on the principle of buoyancy. Upward lifting force is produced by a gas that has a lower density than air, such as hydrogen or helium. This force is calculated from Equation 1 [1].

$$F_B = (\rho_a - \rho_g) g V \tag{1}$$

where F_B is the lifting force, ρ_a is the atmospheric density, ρ_g is the density of the lifting gas, g is the gravity constant, and V is the balloon volume. For expanding volume balloons (refer to Expanding Volume Balloons), the volume of the envelope increases as the balloon ascends to the maximum altitude. Therefore, the balloon will never reach a state of equilibrium, where the forces acting on the balloon (gravity, wind, radiation) equal the buoyant force. For constant volume balloons, the balloon envelope is fixed. Therefore, the balloon is designed to achieve a specific altitude based on the density of the atmosphere at the ceiling altitude e and the mass of the entire balloon system (balloon, payload, string, lifting gas, etc.). Accounting for only forces that affect balloon performance in the vertical direction, the sum of the forces at equilibrium is:

$$\sum F = 0 = F_B - F_g \tag{2}$$

where F_B is the buoyant force, and F_g is the force due to gravity. Substituting Equation 1 into Equation 2 with the definition of the force due to gravity gives:

$$(\rho_a - \rho_g) gV = mg \tag{3}$$

where gravity can be eliminated from both sides of the equation. Equation 3 is solved for the volume required to reach the ceiling altitude in Equation 4.

$$V = \frac{m}{(\rho_a(h) - \rho_g)} \tag{4}$$

The density of air, $\rho_a(h)$, is a direct function of the ceiling altitude, h. Extensive data and models are available relating atmospheric density to altitude.

Zero pressure and super pressure balloons are not filled to maximum volume with lifting gas when launched. This causes the balloon to have a large rate of ascent, and the balloon will pass the target altitude because of momentum. Doing this with a zero pressure balloon will cause lift gas to vent from the balloon and reduce performance. For super pressure balloons, this will cause an increased pressure on the balloon membrane, requiring thicker balloon material. Therefore, the amount of payload mass that the balloon can carry is decreased.

Balloon Types

There are two types of scientific balloons: expanding volume balloons and constant volume balloons. Volume is a reference to the balloon envelop (or the inside of the balloon).

Expanding Volume Balloons

Latex, also called weather or sounding balloons, and rubber balloons are expanding volume balloons. They are made from a membrane that stretches. As the balloon increase in altitude, the pressure from the lifting gas inside the balloon increases as a result of gas expansion. With no means to vent the lifting gas, hydrostatic pressure, and therefore stresses, on the membrane increase. This results in an expansion of the balloon envelope. Eventually, the weakest part of the balloon will experience more displacement than the material can withstand and will tear. The gas on the inside of the balloon will rapidly escape through the tear and cause the balloon to burst.

An example of a flight profile of an expanding volume balloon is shown in Figure 7. The balloon climbs to a maximum altitude, bursts, and then descends. These balloons tend to be less expensive and are disposable. Additionally, a feature of expanding volume balloons is that there is no need for additional flight termination equipment, which is an FAA requirement for constant volume balloons to end the flight. Typical scientific latex balloons can expand to more than 4 times their deflated volume [12].

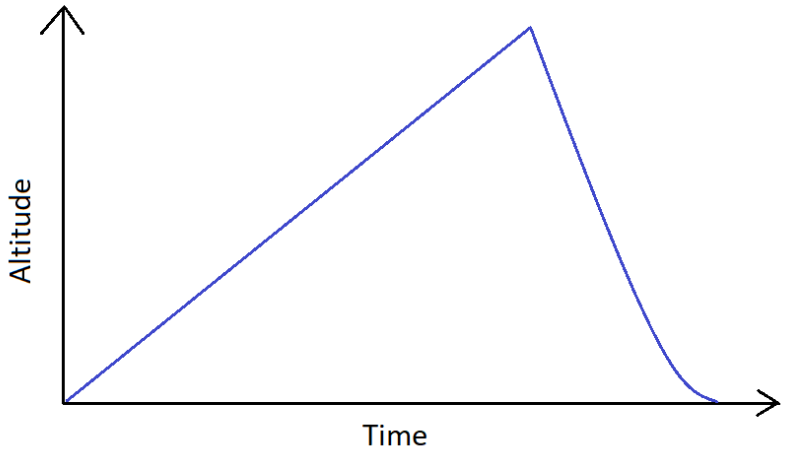

Figure 7: Example Flight Profile of Expanding Volume Balloon

Constant Volume Balloons

Zero pressure and super pressure balloons are considered constant volume balloons. The volume of these balloons' envelopes will remain the same or increase minimally (due to pressure build up for the case of the super pressure balloon). Both zero pressure and super pressure balloons are designed to have a sufficiently large envelope to achieve a specified altitude based on the principles of buoyancy that are discussed in Buoyancy. Both types of balloons are filled partially with sufficient lifting gas to ascend to the desired altitude. Constant volume balloons have flight profiles that allow the balloon to reach a specified altitude and then level off for a duration of time, as illustrated in Figure 8. However, the balloons will only remain at a constant altitude so long as the temperature remains constant. When the temperature drops, for example at night, lift gas in the balloon will compress (increased p), and the volume will decrease, as shown in Equation 5.

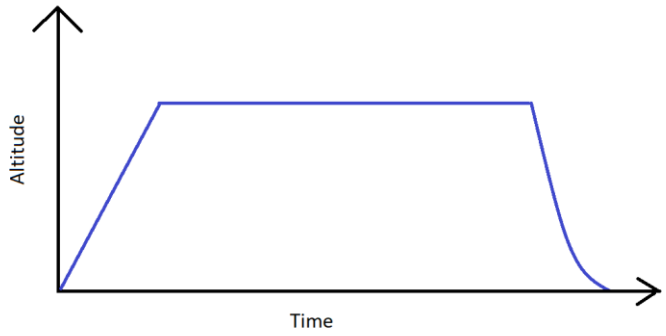

Figure 8: Example of Flight Profile of Constant Volume Balloon

$$pV = nRT \tag{5}$$

where T is temperature, p is pressure, n is the number of moles, R is the universal gas constant (8.3145 J/mol K), and V is the volume. When the volume decreases, the balloon will fall, according to Archimedes principle. As a result, super pressure balloons will oscillate about a desired altitude, and zero pressure balloons will descend until ballast is released. This can be seen from Figure 9, where the zero pressure balloon is below the super pressure balloon. The shaded regions of the image represent night, and lighter sections represent day.

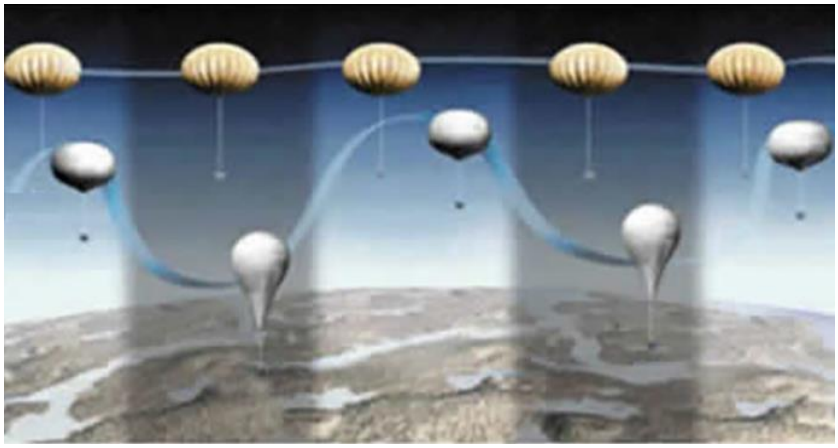

Figure 9: Night Time and Day Time Cycle of ZP and SP Balloons [20]

Super pressure balloons differ from zero pressure balloons in that they are sealed after the balloon is filled with sufficient lift. As a result, super pressure balloons do not vent lifting gas once the envelope is fully inflated. As the name suggests, once the lifting gas expands to fill the balloon envelope, there will be an increase in pressure until the balloon reaches the ceiling altitude. These balloons require a thicker membrane than a zero pressure balloon and utilize stronger load tape to distribute hydrostatic loading from the membrane. Additionally, the load tape must support payload weight. Super pressure balloons are also referred to as pumpkin balloons because when the balloon achieves maximum internal pressure, the wall of the membrane bulges out between lines of load tape, creating a pumpkin-like look. Due to the sealed aspect of super pressure balloons, there is minimal loss of lifting gas, but permutation of gas through the membrane can still occur. Super pressure balloons from NASA have been known to stay aloft for over 100 days [13].

Like super pressure balloons, helikites are super pressure balloons but remain at a low altitude and don't allow for gas expansion to cause the balloon to burst. Helikites are used as tethered systems when there are high winds.

Zero pressure balloons are open to vent excess lifting gas out of the balloon once the envelope is fully inflated at the ceiling altitude. This causes the pressure on the inside of the balloon to equalize to the pressure of the surrounding atmosphere, resulting in minimal stress on the membrane from gas expansion. The payload weight must still be supported by the load tape, but the membrane can be

significantly thinner than what is required for super pressure balloons. Although these balloons can maintain altitude for several hours, they lack the ability to stay aloft as long a super pressure balloon (with similar size, mass, and volume).

Related Experiments

In this section, previous research in the area of biological air sampling and different methods for collection are discussed. In 2014, a research team at Louisiana State University (LSU) used latex balloons to sample microbial aerosols from high altitudes [14]. Their sampling system achieved altitudes up to 38 km using a 2000-gram latex balloon. The total mass of payload suspended below the balloon was 5.4 kg. They used the following equipment to perform the experiments: Trimble Copernicus II global positioning system (GPS) receiver, Byonics Micro-Trak RTG FA Automatic Packet Reporting System (APRS), Kodak Zx1 HD Camera, Arduino Mega 2560 microcontroller, microSD card writing shield, four linear actuators, and 40 Rotorods for the sampling of the aerosols. This work showed that the sterilized sampler that was launched with the balloon did not vary significantly from a sterilized control that remained on the ground [14].

In 2002, researchers from India used cryosampler assemblies attached to latex balloons that achieved altitudes of 20 to 41 km. The goal of these experiments was to show evidence that there were living microbial cells at these high altitudes. In the cryosampler, 0.45 μm microspore filters were used to capture microbes that were later examined. Results showed that clumps of living bacterial cells are in the atmosphere between the altitudes at which they sampled at [15].

Researchers from France in 2005 sampled clouds near Puy de Dôme in Massif Central, France. The mountain has limited access to cars and the researchers walked 5 km to the summit to prevent contamination. During the months of November through March clouds form right over the summit of this mountain. They used single-stage cloud collectors to sample cloud droplets and observed microbes from them. Results show that microbes were found in cloud water [16].

From January 2012 through March of 2013, researchers from the University of Colorado sent out sampling kits to all 50 states plus District of Columbia to participants to sample their exterior doors [17]. The sampling kit were sent in the mail to 1,430 participants with instructions, and dual-tipped sterile BBL CultureSwabs. Once the participants collected the sample from the top of an exterior door, they were shipped back to the researchers and stored at -20°C until they were processed. This research found that there was a correlation between bacteria and fungi with different environmental factors.

To look at the relationship between airborne bacterial community composition and the amount of vegetation in urban areas, researchers from the University of Oregon sampled 5 pairs of parks and parking lots around Eugene, Oregon [18]. They used small vacuum pumps connected to 3 button samplers with filters to trap the microbes, and 3 passive settling dishes. Their sampling was held on July 24th, 2013 and lasted for 8 hours. Once the samples were collected along with temperature, relative humidity, wind direction, and wind speed. No significant

difference was found between parks and parking lots, but they collected between 379,687 and 721,208 sequences using this method of sampling.

Researchers form Georgia Institute of Technology in Atlanta, GA, used NASA's DC-8 platform to sample cloud and cloud free air masses over the Caribbean Sea, and in two tropical storms, Earl and Karl [19]. The DC-8 platform is a modified Douglas DC-8 jetliner that is used as a flying science laboratory. Vacuum pumps connected to Whatman cellulose nitrate membranes filters that were connected to the outside of the DC-8 airplane were used to collect samples. The results from these experiments suggest that the microbes that were collected possess traits that allow them to live in the troposphere.

In 2008, a study was done on stratospheric microbiology at 20 km over the Pacific Ocean using modified impactor plates [20]. These impactor plates were mounted on the wing tips of a NASA modified Lockheed Martin ER-2 and were pressured sealed to remain closed until the pilot triggered them to open. This research was performed on April 28th, of 2008 and the plane flew at an altitude of 20 km. The samplers remained open for 7.5 hours. The results found from this research showed that there are microbes that are in the stratosphere over the Pacific Ocean. They mention that it since the external surface of the sampler was exposed throughout the entire flight, contamination of the low altitude microbes may have been seen in the results.

CHAPTER II

SAMPLING VESSEL

Design

To collect AMCs, several sampling methods were considered, including petri dishes, sticky ribbon tape, and a small vacuum pump that uses button samplers with membrane filters. After several ground level experiments were performed by biology researchers, the petri dishes and the vacuum pump were found to produce similar results. The vacuum pump is approximately 2 lbs and has unreliable programming features and battery life. Therefore, petri dishes are the chosen sampling method for the project. Square petri dishes are being used rather than conventional circular dishes because they have a higher packing coefficient. Since no published information exists discussing the surface area required to obtain sufficient DNA for atmospheric microbiome sampling, the surface area from an original ground-based benchmark experiment conducted by the biologists was used: 20 circular petri dishes. By changing to square petri dishes, only 16 are required to have an equivalent surface area as the circular dishes.

The sampler box was designed to be compact and light weight, and the biology research team required maximum air flow over the petri dishes. Several ideas were developed including using a spiral array, fixed petri dishes in multiple orientations, a circular array, and an expanding array.

It was thought that a spiral array would allow the petri dishes to be placed at different heights on the sampler and allow for different orientations. Due to the complexity of trying to keep the petri dishes protected from contamination, this idea was developed into an accordion-type structure. The accordion structure expanded in such a way that the petri dishes would experience a different orientation of the air with each layer as it was expanded. A prototype of this idea can be seen in Figure 10. Experiments with this prototype resulted in tangled strings, and it was very time consuming to ensure the string lengths were the same. This made reproducibility extremely difficult.

Figure 10: Accordion Array Prototype

The final decision was to use an expanding array. With this solution, the bottom of the sampler box lowers so that the expanding array extends significantly further than the height of the top of the box. This maximizes the amount of air flow around and over the petri dishes while keeping the sampling system as simple and compact as possible. The expanding array prototype was tested and can be seen in Figure 11. These experiments were done to ensure that the sampler would collect AMCs.

Figure 11: Expanding Array Prototype

Materials for construction of the exterior of the sampler box were chosen based on several considerations. Two primary considerations are the weight that the material adds to the sampling system and material cost. The sampler box cannot weight more than 6 pounds to be within FAA regulations [21]. Several

sampler boxes are required for the experiment; therefore, the cost per box was minimized. Additionally, the biology team had the following requirements: the sampler box shall be easy to decontaminate and the materials shall be inorganic. Fibrous, and porous materials are difficult to sterilize, and organic materials were restricted to prevent DNA from the sampler box contaminating the collected samples. The material selected for the exterior of the sampler box is 1/32" thick Aluminum 3003 sheet. Aluminum is a relatively lightweight and strong material. Using the 1/32" thick aluminum allows for the box material to be light and capable of bending in a pan break tool without failure due to crack propagation. The 3003 class of aluminum fits these requirements.

The sampler box and its assembly were designed using computer aided drawing (CAD) software, Autodesk Inventor 2017. Each part was created in this software and later assembled in a full assembly drawing. This allowed for virtual visualization of the sampler box. Modifications were made within the software to ensure that there was enough space for moving parts and petri dishes. The CAD model can be seen in Figure 12. Another requirement for the sampler box design was the ability for the box to fit within a UV sterilization hood used by the biology team to prepare the sampler box for sampling.

The sampler box is designed to hold 16 square petri dishes on an extending array. The array is made of G10/FR-4 board, stainless steel wire, and wire crimps. The G10/FR-4 is a composite material commonly used for prototyping circuit boards.

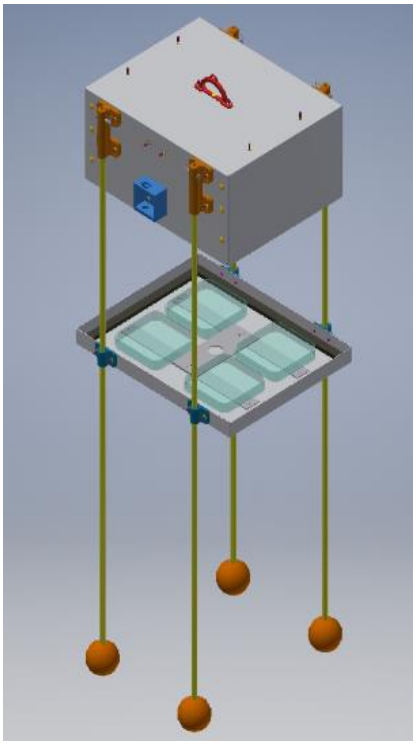

Figure 12: CAD Model of Sampler Box with 4 Tent Poles

This material was selected because of its strength and ability to be autoclaved, a method for sterilizing sampling equipment by heating materials to 121˚C, which is the temperature required to kill bacterial spores. Although this sterilization method is not actively used for the current experiments, having this option for sterilization of the array is a feature that could help with contamination mitigation. A custom UV chamber was constructed to decontaminate sampler boxes with chemical and UV sterilization methods. The chamber allowed for sterilization of a sampler box in remote locations due to its portability. While the sampling box is in the UV chamber, a 70% ethanol mixture is applied to the sampler box walls and other surfaces within the sampler box to sterilize areas that not in direct line of sight of the UV lightbulbs.

Three options were considered for maintaining alignment while lowering the array from the top lid of the sampling box. These methods included the use of 3 telescoping poles, 1 telescoping pole with spring steel for alignment, and aluminum tent poles. The first method tested was to make loosely fitting telescoping assemblies from aluminum tubing and attach them to the sampler box on two corners and through the center of the box, as seen in Figure 13. This would have allowed the sampling box to be stored in a very compact manner. One end of an aluminum tube was beveled to prevent the inner tube from falling all the way out, and the top of each tube was flanged, such that it would be stopped by the bevel. After constructing several of these telescoping poles, the precision required to machine the telescoping poles to easily slide was too high to be a functional solution for the construction of many sampling boxes. Additionally, the telescoping tubes caught on each FR-4 board layer of the array.

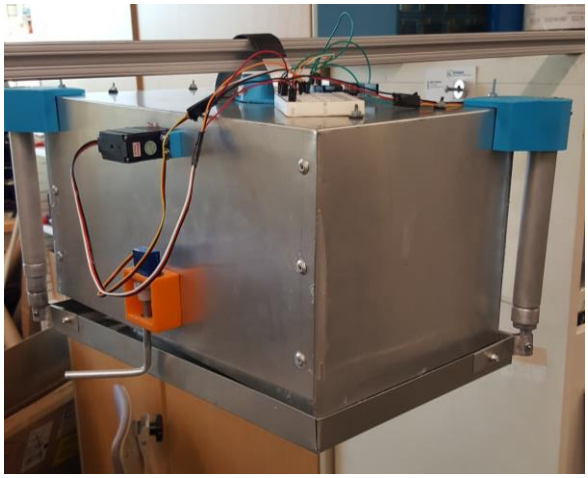

Figure 13: Telescoping Pole Prototype

The next prototype used one telescoping pole in the center of the box, and spring steel from a tape measure on the sides of the box, as shown in Figure 14. As the box opened and closed, the spring steel kept the bottom lid in rotational alignment with the top of the box, and the central telescoping pole prevented the bottom lid from translating too far. Although this system worked, when tested on the aerostatic helikite, the tossing and turning of the box in the wind caused the tape measure to become twisted and rip from the bottom lid. Therefore, the sampling box did not close properly.

Figure 14: 1 Telescoping Pole Prototype

Finally, a sampler box that uses 4 aluminum tent poles, one at each corner on the box, was made. A 3D printed part, the top tube collar, secures the tent poles to the top of the box. The collar uses a clevis and cotter pin to prevent the tent pole from falling out. The bottom lid has a sleeve through which the tent poles slide freely. The bottom sleeve is 0.12 in larger in diameter than the 0.33-in-diameter

tent pole to maintain alignment with the top lid. The bottom sleeves are 2 in long and prevent the bottom lid from kinking. This sampler box is shown in Figure 15.

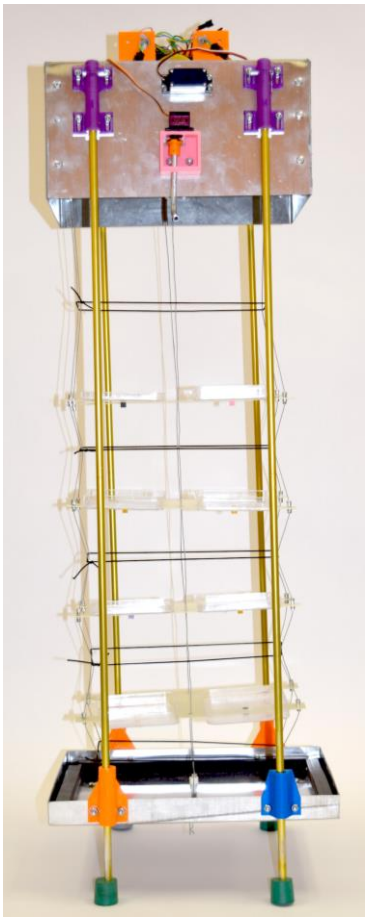

Figure 15: Final Sampler

Further improvements have been made on the sampler box. These include: making the bottom tube sleeves taller to prevent them from getting caught on the tent pole; increasing the size of the bottom lid to prevent misalignment and ensure proper sealing; replacing foam on the bottom of the tent poles with rubber stoppers; and adding elastic cord between each array layer to ensure the wires are folded toward the center of the box as it is closing. These improvements are all reflected in the sampler box shown in Figure 15. Additionally, the 3D printed parts were

improved to either provide better functionality, provide improved print quality, or reduce weight.

Sampler Box Functionality

The sampler box is designed to lower the expanding array using a continuous rotation servo motor that controls a crankshaft. Stainless steel uncoated wire is used to connect the crankshaft to the bottom lid. For the tethered balloon system, communication between the sampler box and the ground occurs between two Arduino Mega 2560 microcontrollers. These microcontrollers are outfitted with a wireless 433 MHz frequency communication chip called an HC-12. One microcontroller is used as a remote control operated by a user on the ground. On the remote, the user presses a momentary button, and the remote sends a unique signal that correlates to a unique command on a specific sampler box. These commands begin and end the sampling operation, open and close the locking arms and cause the crankshaft to reel up and down for one second. The "begin sampling operation" commands two servo motors to rotate the locking arms from a closed position where they prevent the sampling box from opening, to an open position. Once the locking arms are rotated, the expanding array rolls down to expose the petri dishes.

When the sampling box is opened, the onboard microcontroller performs a software reset that returns the processor to an idle state. This idle state awaits another HC-12 communication signal while collecting temperature, pressure, and humidity data using a BME-280 sensor from Adafruit. The data are stored on an SD

card using an SD logging shield (also from Adafruit) with an internal clock that time stamps the data. If the microcontroller on the sampling box does not receive a signal from the remote control after 8 hours of sampling, it is automatically programmed to carry out the “end sampling operation.” The "end sampling operation" retracts the expanding array until a limit switch tells the microcontroller that the box is closed. Then, the locking arms are rotated a closed position. The “end sampling operation” can also be triggered from the remote.

In the event the box does not completely close, two additional momentary buttons can be used on the remote control to open the locking arms, turn the crankshaft up for one second and close the locking arms. The “one second” momentary button will also cause the crankshaft to reel down for one second in the case the sampler box does not open completely. These additional sampling box control features were added after some experiments resulted in the expanding array not rolling up or down completely, causing the locking arms to interfere with the bottom lid when the box was further closed. With the open/close lockout arm commands, the ground based operator can open the lock out arms, and then use the one second up command” to ensure the box is completely closed. The locking arm open and close operation is also beneficial in testing that the sampler boxes are turned on and receiving a signal prior to attaching them to the balloon.

For the tethered sites, there are 3 sampler boxes per site. The remote control is set up in such a way that it can control the 3 sampler boxes at the testing site by sending unique codes to each box individually. Additionally, there are 3 sites per

sampling location that may be as little as 1 km apart. The HC-12 wireless communication chip is capable of an operating range of up to 1.2 km. To prevent any cross communication between sample sites, each controller and sampler box are programmed uniquely to read/send codes specified for an individual sampler box. This ensures that only sampler boxes at the designated site will open when a command is sent. A picture of the remote can be seen in Figure 16.

Figure 16: Remote for Tethered Sampler Boxes

Each sampler box has three servo motors. One is a high torque continuous rotation servo, and the other two are high torque limited range servos. All servos have metal gearing and shafts, and each servo requires a coupler to connect the shaft of the servo to an aluminum tube. The continuous rotation servo is used as the crankshaft motor. This servo can complete multiple revolutions allowing the servo to lower and retract the expanding array based on the amount of time set in the

microcontroller coding or the command that the microcontroller receives from the remote control. This servo has an aluminum coupler that is made from 1/2-in-diameter AL 6064 rod. The holes in the sides of the coupler are used to make mechanical connections. This coupler connects the shaft of the servo to the 1/2-in-diameter 3003 aluminum tube crankshaft.

The limited rotation servos are motors for the locking arms that have a 3D printed plastic coupler connecting the servo shaft to a 1/4-in-diameter aluminum tube. This aluminum tube is bent to provide a lock so the box's bottom lid remains sealed shut when not sampling, preventing accidental contamination of the sterilized box or collected samples. A spring pin is used to make the connection between the lockout motor coupler and the aluminum tubing to prevent the tubing from falling out of the coupler.

All servos are powered by a 7.4-V 1000-mAh 2-cell lithium ion polymer battery. Similarly, a separate 7.4-V 2200-mAh 2-cell lithium ion polymer battery is used to power the onboard microcontroller. Each of the batteries have been altered with a microfit connector that allows for a secured connection to power the microcontroller and servos. Fully charged batteries are desired for all samplings; therefore, rechargeable batteries reduce cost over time.

Unlike the tethered system, the launched Arduino Mega system is programmed to open and close automatically. The opening operation is triggered by using the pressure that the BME 280 sensor reads. When the pressure read by the Mega, is less than or equal to the pre-programed pressure associated with the

desired testing altitude, the box will open. If the balloon never reaches the target altitude or the pressure sensor fails to read correctly, the sampler box will open after one hour of flight. The box will remain open, sampling for 8 hours and close automatically. Additionally, if the sampler box must be closed early, a command can be sent to a dual tone multi-frequency (DTMF) decoder using HAM Radio signals. This command activates a code on the onboard microcontroller that will close the box and perform a cut down operation that will bring the balloon down to the ground. The cutdown feature is discussed in the section Cut Down Devices.

The Arduino codes for the tethered sampler box and remotes can be seen in Appendix A: Arduino Code for Tethered System and Appendix B: Arduino Code of Tethered System. Additionally, the Arduino codes for the launched sampler box and Arduino mini for the cut down device can be seen in Appendix C: Arduino Code for Launched Sampler Box and Appendix D: Arduino Code for Launched Tracker Box.

CHAPTER III

TETHERED SYSTEM

A tethered helikite system was used to perform sampling at 30 m and 150 m above the ground. The tethered system consists of an aerostatic vehicle that is anchored to the ground.

Latex Balloons

The first attempt at developing the tethered system used a latex weather balloon connected to 100 lb break test fishing line. The balloon was filled with helium to lift the sampling payload. The primary problem experienced during this first attempt was that a balloon alone did not provide sufficient stability at high wind speeds. To provide lift in stronger winds and increase stability, the next system used a chambered kite structure suspended below the balloon. Stability in this context means the helikite doesn't sway vigorously in the wind. The goal of the kite was to produce lift in high winds. The kite was constructed from PVC tubing and monokote, a film that is typically used on remote controlled airplanes. During testing on a windy day, the chambered kite did not provide sufficient stability to keep the payload aloft. The kite needed to be attached to the balloon rather than below it because the balloon and the kite pulled in opposite directions. Furthermore, the lifting surface needed to be larger than what was used with the chambered kite.

Helikite

A helikite is an aerostatic vehicle that uses a balloon filled with lifting gas attached above a kite section to produce lift and stabilization in windy conditions. After designing and constructing a helikite in house, the time commitment and resources required to be successful outweighed the cost of a commercial-off-the-shelf helikite. Allsopp's 16 m^3 helikites, shown in Figure 17 and Figure 18, are used as the tethered aerostatic vehicle. Table 1 shows all of the tethered tests throughout the project. The final six tests were sampling field experiments for the biology team.

Figure 17: Close View of Tethered Helikite While in Flight

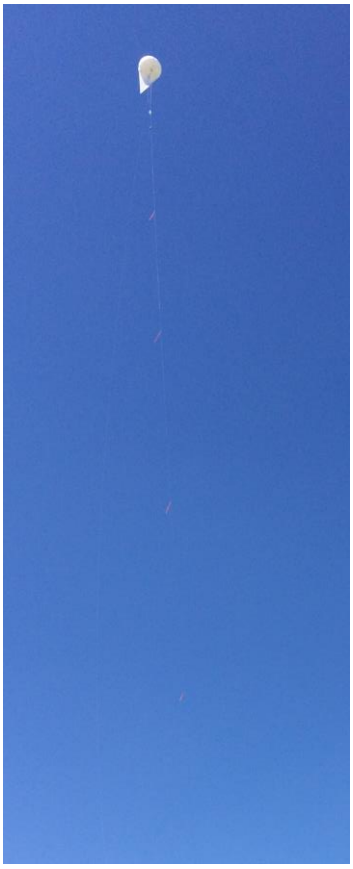

Figure 18: View of Tethered Helikite with Sampler Boxes

Table 1: Summary of Tethered System Tests

Date	Test	Duration	Comments
11/1/2015	Latex Balloon With Kite- MonoKote	1 minutes	Fail – insufficient stability and lift
2/6/2016	Plastic Balloon Homemade Helikite - Mylar	2 minutes	Fail – Mylar material failure
2/7/2016	Plastic Balloon Homemade Helikite - Cloth	5 minutes	Fail – kite too loose
3/8/2016	Plastic Balloon Homemade Helikite - Cloth	5 Minutes	Fail – kite too loose - Tethers Broke
5/6/2016	Helikite 12 m^3 with Dummy Loads	4 Hours	Fail – in sufficient lift
5/24/2016	Plastic Balloon with Array	4 Hours	Green Balloon - No box, just plates
6/24/2016	Helikite 16 m^3 with Prototype Sampler	4 Hours	Telescoping Prototype
10/15/2016	Helikite 16 m^3 with 2 Prototype Sampler	6 Hours	Spring Steel Prototypes – Spring steel broke
12/20/2016	2 Helikites - Grand Rapids	2 Hours	Windy
12/21/2016	2 Helikites - Kalamazoo	6 Hours	Finalized Samplers - two helikites
12/23/2016	3 Helikites - Kalamazoo	6 Hours	Finalized Samplers - All 3 helikites
1/15/2016	3 Helikites - Pellston	6 Hours	Successful
2/11/2017	1 Helikite - Kalamazoo	6 Hours	Successful
3/23/2017	3 Helikites - Kalamazoo	6 Hours	Successful
5/3/2017	Jornada	6 Hours	Successful
5/6/2017	Albuquerque	6 Hours	Successful
5/11/2017	Denver	6 Hours	Successful
5/13/2017	CPER	6 Hours	Successful
5/17/2017	Kalamazoo	6 Hours	Successful
5/19/2017	Pellston	6 Hours	Successful
5/24/2017	Harvard Forest	6 Hours	Successful
5/27/2017	Boston	6 Hours	Successful

CHAPTER IV

LAUNCHED SYSTEM

The launched balloons are designed to hold the sampler box at 5000 meters above sea level. As discussed in Chapter 1, an expanding volume balloon will not maintain a specified altitude and therefore a constant volume balloon was used for this research. Additionally, zero pressure balloons were selected because they can maintain altitude, terminate the balloon flight by venting lifting gas, and are easier to manufacture when compared with super pressure balloons.

Manufacturing a Zero Pressure Balloon

Initial attempts to make a zero pressure balloon used a MATLAB code from Purdue's AMET website [22]. This website provided insight on how to design, and make a zero pressure balloon from painter's tarp, packaging tape, and a hair straightener. The painter's tarp is a 0.9 mil thick polyethylene. The MATLAB code inputs are: payload weight, desired altitude, number of gores, and lifting gas. The tracker box, discussed in the section Tracking Devices, is approximately 4.2 lb and sampler box is 5.21 lb. Other attachments to the launched balloon include a radar reflector (0.8 lb), load ring (0.05 lb), and strings to attach all payloads (~0.1 lb) to the balloon. For the initial design, these other items were approximated to have a weight of 1 pound. The Matlab code calculates the approximate weight of the balloon, so it is not required as an input. The altitude was set to 5 km, the number of gores was chosen as 8, and helium is the lifting gas.

A gore is a section of the balloon; gores are fused together to make the balloon. The number of gores selected is important for the performance of the balloon. When more gores are used, less stress is experienced by the load tape. However, aspects such as the weight, construction time, and probability of a leak increase with more gores. The minimum number of gores a balloon can have is 2, creating an envelope. More gores allow the balloon to achieve a natural shape, discussed in Chapter 1, which allows for more payload to be launched. Eight gores were used as an initial input to the Matlab code to determine the size of the gores.

With the given inputs, the Matlab code outputs: gore shape, gore dimensions, balloon full volume, and balloon volume at launch. For the 5 km altitude balloon, these parameters are shown in Table 1.

Table 2: 5 km Zero Pressure Balloon Parameters

Parameter	**Value**	**Units**
Surface Area	16.57	m^2
Volume SI	6.19	m^3
Volume ENG	218.55	ft^3
Mass	0.30	kg
Payload Mass	3.63	kg
Total Weight	38.51	N
Volume of Lifting Gas at Sea Level	131.33	ft^3
Arclength of Gore	3.64	m

A 5 km altitude balloon gore template was made from coordinates produced by the Matlab design code. These values can be seen in Appendix E: Balloon Template Values. The gore template is made by taping paper to the ground and plotting each point generated by the code. This template can be seen in Figure 19. Packaging tape is taped to the gore template to prevent damage to the paper. The polyethylene plastic tarp is placed over the template, and a PVC tube is used to force air out from underneath the plastic. Box cutters are used to cut the plastic in the shape of the gore. This is repeated for the designated number of gores required to make the balloon.

Figure 19: Gore Template

The gores are fused together using heat. This is accomplished with a hair straightener that has a temperature control feature. The reason for using a hair straightener over other heating tools is because it has two heating surfaces and provides sufficient surface contact for the desired 1-inch seam that runs along the edges of each gore. The ideal temperature for the straightener is 275°F. When all gores are fuse together, the balloon is turned inside out so that the sealed seams are on the inside of the balloon. Then the outside of the seams are taped over with packaging tape. The tape acts as a form of "load tape," preventing seam failure. Packaging tape is lighter than duct tape and has relatively good adhesion to polyethylene.

A cross stitching ring serves as a load ring that is attached to the opening of the balloon. It also allows for a location to attach the payloads to the balloon. Cross stitch rings use a screw with concentric rings to secure fabric in place. This clamp is used on the balloon by putting the balloon material around the inner ring, and the outside clamping ring is tightened over the balloon plastic with the screw. The payloads are also attached during this process by tying the connecting string around the load ring. This is shown in Figure 20. The weight of the payloads is transferred through this ring to the balloon tape and membrane plastic.

Tracking and Ending Flight

To recover the AMCs collected form the launched sampling box, a tracker box was developed. The tracker box has two major tasks: providing the location of the balloon, and ending the flight of the balloon.

Figure 20: Load Ring

Tracking Devices

The location of the balloon is tracked with several different devices. The first is an automated position reporting system (APRS). An APRS tracker uses a global positioning system (GPS) modem that receives the latitude, longitude, and altitude from GPS satellites. It transmits that information using amateur HAM radio frequencies. These packets of information are transmitted on 144.390 MHz, which is the US frequency for APRS tracking. Packets can be seen by using a receiving radio with APRS receiving capabilities or through the APRS online tracking network through the use of internet or cellular service at aprs.fi. The capability of tracking with a HAM radio that can decode APRS packets is necessary for when the balloon travels over regions without cellular service. The APRS tracker, Byonics Micro-Trak RTG 10Watt VHF Transmitter, uses a 2-meter bipolar antenna, and requires 8 AA batteries.

A Foxhunt transmitter, Byonics Micro-Fox MF-15, is another tracking mechanism in the tracker box, that uses the same principles as radio telemetry. This transmitter sends a series of tones through HAM radio frequencies for a person with a hand-held HAM radio receiver to track. The tones can be heard up to 2 miles using standard, inexpensive HAM radios. A technique called "body shielding" is used to determine the direction of the foxhunt transmitter. This technique requires the person tracking the foxhunt device to turn in a circle while listening to the tones from the foxhunt device through the receiving radio and finding where the signal is the weakest. Once determined, the foxhunt transmitter will be in the opposite direction of the where the person is facing. As the tracker gets closer to the foxhunt transmitter, the HAM radio receiver frequency may need to be changed from the transmission frequency by a few kHz to provide a weaker signal.

The third method for tracking the balloon and sampler box is a SPOT trace GPS. This product is a GPS transmitter that provides latitude and longitude to the SPOT website. It can then be tracked as long as the user has a subscription to the SPOT tracking service. This service requires a connection to the internet, which is not always available in the sampling locations.

Cut Down Devices

Zero pressure balloons, as mentioned previously, will not pop when they achieve their ceiling altitude. Rather, they will float at a relatively constant altitude depending on sun radiation heating, cold or hot pockets of air, permutation of the lifting gas through the plastic, mixing of the lifting gas and atmospheric air, and

leaks in the balloon. Therefore, it is necessary to be able to bring the balloon and payload down to the ground. This is accomplished using two methods on the launched system: a hot wire and a tow line release.

The launched balloon system is assembled as shown in Figure 21. A deflate line connects the top of the balloon to the tracker box. Either the load ring or a duct is open to the inside of the balloon. The goal of the cut-down devices is to cause the balloon to flip upside down, resulting in helium venting through the load ring or duct.

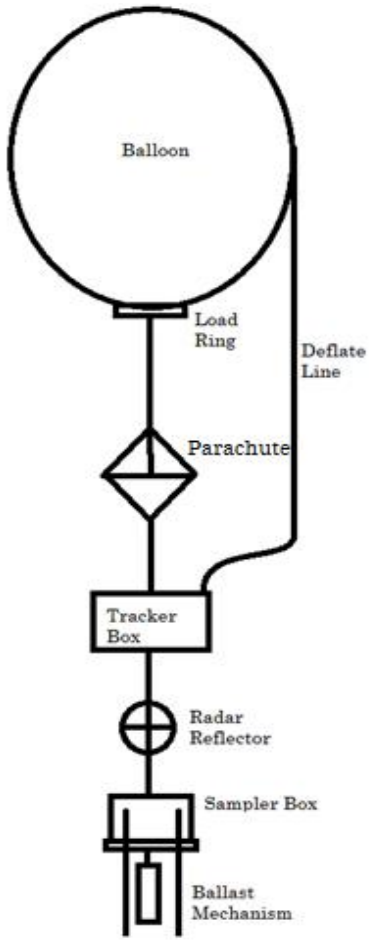

Figure 21: Launch Balloon Diagram

The first cut down method is the hot wire. The programming for this hot wire method can be seen in Appendix D

Arduino Code for Launched Tracker Bo A tungsten wire is wrapped around the line connecting the tracker box to the parachute. The wire is connected to alligator clips. These clips have wires soldered to them that are connected to a 9-V battery pack through a solid-state relay. The relay is connected to an Arduino Mini. The Mini sends a high signal to the relay, which allows the 9-V battery pack to run current through the high resistance tungsten wire. This wire heats up and burns the line, resulting in the deflate line, which connects the tracker box to the top of the zero pressure balloon, being the only string holding the tacker box to the balloon, causing the balloon to invert. The relay is programmed to allow current to flow for 30 seconds.

The tow line release is a mechanism used for remote controlled airplanes that tow glider planes. This mechanism is a mechanical switch that uses a servo motor to release the line. Figure 22*: Tow Line and Hot Wire Schematic* and Figure 23 show a schematic of how the tow line release mechanism operates and an image of the mechanism, respectively. The same line that is burned with the hot wire is directly connected to a stainless steel, plastic-coated wire. This wire is latched into the tow line release. The release is connected to a limited-rotation, metal-geared servo controlled by the same Arduino Mini that controls the hot wire. When the high signal for the servo is sent, it rotates the latch to release the wire. The lift of the balloon, and the weight of the sampler box, and radar deflector will cause the

hotwire line break so that the only line attaching the tracking box to the balloon is the deflate line.

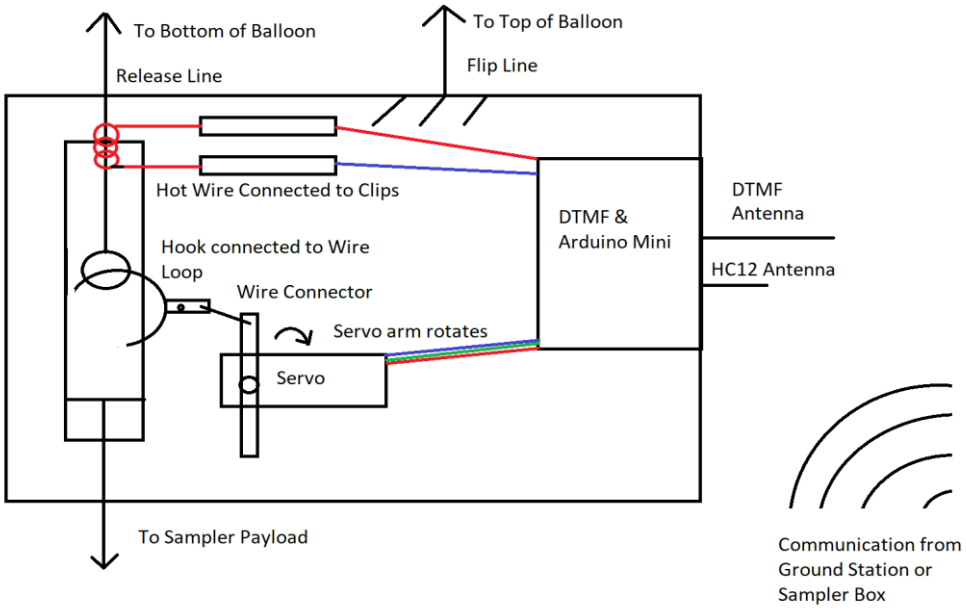

Figure 22: Tow Line and Hot Wire Schematic

Figure 23: Tow Line Release [27]

The ground system that is used to track the balloon and payload consists of handheld HAM radios, cell phones, and a ground station HAM radio with a laptop. As mentioned above, the handheld HAM radios are used for tracking the foxhunt transmitter. The cell phones can be used to track APRS packets using the aprs.fi

website and track the SPOT using the SPOT Trace app or website. These options are only possible where cellular service or Wi-Fi is available. The HAM radio ground station is used for tracking the APRS tracker when there is no cellular service or Wi-Fi availability.

Launched Balloon Troubleshooting

Launched balloons were tested with a dummy payload to substitute for the sampler box. The dummy payload was a foam box that held the sampler box controller and additional mass to simulate the weight of the sampler box. Results from initial zero pressure balloon launches demonstrated that original balloon design would not work. The zero pressure balloon ascended to a ceiling altitude, leveled off for a short time, and then began a slow descent.

To maintain the ceiling altitude for at minimum 6 hours, a requirement set by the biologists, modifications were required for the balloon. These changes included: switching to a cylindrical balloon, adding a ballast system, and adding a venting duct to the bottom of the balloon.

The primary purpose of switching to the cylindrical balloon was to ensure that the manufacturing process was not causing the early descent. With the natural shape balloons from the Purdue AMET code, the seams of every gore must be fused together with heat and then taped over with packaging tape. If the seam is the means by which lift gas is lost, there is increased risk in having many seams sealed by hand. The advantages of switching to a cylindrical balloon is the number of seams is reduced significantly. The primary disadvantages with cylindrical balloons

is that with heavier payloads, there is an excess amount of plastic near the bottom of the balloon. This results in a balloon that is not optimized and heavier than necessary, as discussed the section Buoyancy, Cylindrical balloons can be made by taking two sheets of plastic and sealing around all the edges or by obtaining a tube and sealing the top and bottom. Marshal Plastic from Martin, Michigan donated a roll of polyethylene 1.5 mil thick lay flat tube that is 20 feet in circumference. The roll of flat tube is rolled out to 19.7 feet and cut from the rest of the roll. The top of the flat tube is sealed completely, and the bottom is sealed leaving a 1.6-foot gap where the load ring is placed. Like the natural shape balloons, the cylinder is then flipped inside out and the seams are taped over with packaging tape. A test was done by only switching to a cylindrical balloon and the results remained the same: the balloon achieved a ceiling altitude and slowly descended. Although the results did not suggest that the seams of a natural shape balloon were faulty, the ease of manufacturing the cylindrical balloon led to its adoption for use for the remaining tests.

The next system that was tested is a ballast system that dumps sand when the balloon descends below a specified altitude. When the balloon descends, its lift must be less than the weight of the system. By removing mass from the system (sand in this case), the lift will become greater than the weight and sustain the sampling altitude longer. The ballast system is a 12-in-long, 2-in-diameter PVC tube that holds the sand. 3D printed parts were designed to mount a servo motor to the bottom of the PVC tube. This can be seen in Figure 24. A piece of polycarbonate

is attached to the servo to serve as a door to cover and uncover a hole through which sand drops. The ballast system holds 1.2 lbs of sand that can be dropped during the

Figure 24: Ballast Mechanism

flight. The servo opens and closes the hole in the adapter for ~ 1 second. As a result, the ballast system can drop sand 77 times before it runs out. This system is suspended form the bottom lid of the sampler box, with a control cable that runs through one of the tent poles to connect the sampler box microcontroller with the servo. This can be seen in Figure 25.

The ballast system acts as method to extend the flight time of the balloon, but does not address the problem of why the balloon is descending early. Further investigation of this issue found that at lower altitudes, air and helium do not mix easily, but at higher altitudes this is not the case [2]. The method that is used to prevent air and helium from mixing at higher altitudes is a collapsible venting duct

attached to the side of the zero pressure balloon in place of the hole at the bottom of the balloons previously tested. To add this to the cylindrical balloon, the opening

Figure 25: Ballast Connected to Sampler

where the load ring is attached is sealed and a plastic tube is attached to the side of the balloon. This tube extends past the bottom of the balloon so that when the helium expands in the balloon, it will fill the entire balloon envelope first before venting out of the tube. When the balloon is ascending, the tube is collapsible and mitigates the amount of air that can get into the envelope of the balloon. When the balloon achieves ceiling altitude, the lifting gas will have expanded, filling the bottom of the balloon and escaping through the duct, allowing the balloon to remain a zero-pressure balloon.

CHAPTER V

FUNCTIONALITY EXPERIMENTS

To ensure that the sampler boxes, tethered balloon system and launched balloon system will provide the information that the biology team needs, several experiments were performed.

Testing the Sampler Box

Numerous experiments were performed on the sampler boxes including the ability to collect samples, contamination experiments, cold weather functionality, 8-hour automatic end sampling, simulating high wind conditions, 3D part quality, and impact testing.

Collection of AMCs and Contamination

The primary objective of the sampler boxes is to collect AMCs at the desired altitudes. The biology team conducted several experiments to ensure that the sampler boxes collect AMCs. These experiments tested sampling duration, petri dish charge state, and collection substrates. The results of these experiments were that an 8-hour sampling time on an uncharged petri dish coated with silica gel were the ideal sampling conditions to allow for detectable amounts of AMCs.

Another experiment was performed to compare microbial collection between an open array, an open sampling box, and a closed sampling box. The biology team wanted to ensure that the lid of the sampler box was not hindering the collection of AMCs by preventing flow between the layers of the array and shielding the petri

dishes. Additionally, the sampler boxes are designed to prevent contamination. When the box is closed, it should be sealed tight enough to prevent contamination of the petri dishes. The amount DNA extracted using each technique is shown in Table 3.

Table 3: Sampler Box Collection and Contamination Test Results

Subject	Value	Units
Open Sampler Box	164	ng/mL
Closed Sampler Box	Too Low to Detect	ng/mL
Open Array	162	ng/mL

The tests show that the open sampler box collected as much DNA as the open array, ensuring that the top of the sampler box is not preventing the collection of AMCs. Additionally, the closed sampler box prevented DNA from getting on the petri dishes.

Functionality and Hallway Testing

Every time a sampler prototype or final design sampler box was made, tests were performed to check the functionality of the Arduino commands and box deployment and retraction in simulated wind.

The functionality tests included opening and closing the locking arms, and ensuring that they sealed the foam in the bottom of the sampler box against the top lid. This test also checked that the lock out aluminum arms were attached in the correct orientation. Another functionality test was performed to check the timing of

the crankshaft servo. The servo motor used for the crankshaft is a continuous rotation servo that turns for a specified time. The controller box is programmed with the time to allow the array to be fully extended. Additionally, the box close time is adjusted to ensure the crankshaft servo operates long enough to close the box, but not too long that it will break the servo motor. After further improvements of the sampler box, a limit switch was added to stop the crankshaft servo from turning to close the box when it was compressed. This prevented the servo from breaking. The final functionality test was to ensure the one second up and one second down commands worked. on the sides of the boxes, and turning the crankshaft up and down for one second. The same tests were performed with the remote control 150 m from the sampler boxes to simulate the distance that the remote will be from the highest tethered sampler box

Another experiment tested the sampler box operation for the expected 8-hour sampling time to ensure that the microcontroller code automatically closes the box and the batteries have sufficient power to function. This same time test was performed at cold temperatures to ensure the microcontroller system and batteries function at lower temperatures for winter sampling. In addition, a hand warmer was put inside of the control box to keep the microcontroller and batteries warm. Finally, an experiment was performed to ensure that weather data recorded by the microcontrollers are consistent and correct.

After these functionality tests were completed, the sampler box was moved to an area that provides sufficient room to swing them around in vigorous motion to

test for necessary ruggedness of the box in windy conditions While the sampler boxes were spun and pushed, the same functionality tests previously discussed were performed.

Testing the Tethered Balloon System

The tethered system uses an aerostatic vehicle called a helikite, which is a product from Allsopp Company. Helikites are balloons with a kite structure attached that provides additional lift with increased wind speeds and improved stability in comparison to a simple tethered balloon.

During an early sampling, the helikites struggled to maintain altitude as the temperature increased throughout the day and in low wind situations. The loss of lift due to decreased air density caused the 30-meter altitude sampler box to touch the ground in one sampling location. An investigation of the amount of lift produced

Table 4: Helikite Lift Calculations

LOCATION	ELEV	PRESSURE	AVG. TEMP		MASS OF HELIUM		BUOYANT FORCE		EXCESS LIFT	
			Avg	High						
[-]	[m]	[Pa]	[K]		[kg]		[N]		[N]	
Boston	9	101000	288	292	2.71	2.67	191.30	188.33	25.33	22.74
UMBS	233	98815	285	293	2.67	2.60	188.59	182.79	23.03	17.95
Kalamazoo	290	98070	288	294	2.62	2.57	185.00	180.68	19.90	16.11
Harvard	332	97562	287	293	2.62	2.57	184.75	180.45	19.70	15.93
Jornada	1428	85608	293	301	2.25	2.19	158.14	152.97	-3.30	-7.89
Albuquerque	1619	83136	291	298	2.20	2.15	154.78	150.39	-6.17	-10.06
Denver	1639	83136	285	292	2.25	2.19	158.51	154.15	-2.90	-6.72
CPER	1645	83272	286	293	2.24	2.19	158.14	153.78	-3.22	-7.05

by the helikite was performed to determine whether this problem would continue at sampling sites located at higher elevations than Kalamazoo, MI. These calculations

(found in Appendix F: Balloon Lift Calculations MATLAB Code) required an approximate value for the mass of the entire suspended system from the helikite. The mass was determined by adding the mass of the sampler boxes, the helikite poles, helikite, SPOT Trace, and the tethering line. The total weight was found to be approximately 139.36 N. Equation 1 was used to determine lifting force of the helikites. Table 4 shows input parameters and calculated expected lift at various sampling locations across the United States for May 2017 sampling. Input values include: elevation, average air pressure in May, average daily temperature in May, and average high temperature in May. The average high temperature results and values are in the shaded columns. Additionally, the table shows the amount of helium, in kg, required to produce maximum lift from the 16 m^3 helikite, assuming no wind. The weight of the system is subtracted from the buoyant force resulting in excess lift. Negative values of excess lift mean that the helikite is not capable of lifting the sampling system, and these occur at higher elevations and warmer temperatures. The most severe case is Albuquerque, NM where the elevation and average high temperature would result in the need of over 10 N more lift than the maximum that can be provided by the helikite to lift the entire sampling system.

Two options for solving the problem of insufficient lift were considered. The first option was to reduce weight of the sampling system. Several 3D parts at reduced infill densities and reduced build volumes were produced and tested to determine the impact on the sampler box. This option would significantly decrease

the strength of the 3D printed parts and adversely impact the survivability of the sampler boxes.

A second option is to increase the buoyant force by increasing the lift. This was done using a technique called “tandem ballooning”. Tandem ballooning is a method in which two or more balloons are connected to produce more lift. To incorporate this idea with the helikite, a PVC advertising balloon was attached to the tether line of the helikite between the sampler boxes. This solution is only implemented in low wind conditions because the attached advertising balloon does not have the stability of a helikite in high winds. Furthermore, the helikite produces sufficient lift in high wind conditions where the kite increases the lift. A picture of the tandem balloon configuration is shown in Figure 26.

Testing the Launched System

Due to the complexity and chance of error of the launched balloon system, several experiments were done to ensure the system performed properly.

Tracking System

The tracking equipment consists of a SPOT trace, foxhunt device, and APRS tracker (all discussed in the section Tracking Devices). The SPOT and APRS trackers were tested during a tracking training performed by members of the team. Two members of the team had the trackers and the remaining team tracked them. This training/test determined that the APRS tracker can give a location if it is captured by an APRS receiving HAM radio and laptop, or through local repeaters.

Figure 26: Tandem Tethered Helikite Test

The SPOT tracking device was found to work well when it remained powered on. The device will turn off automatically if it is still for too long. The foxhunt was tested several times by hiding the device and using hand held HAM radios to find it. Additionally, there was an event when a zero pressure balloon broke free of the tether. With only the foxhunt sending a signal, the means of tracking the system was done through the body shielding technique previously described. The combination of all three devices has been a reliable means of tracking through numerous testing and training exercises.

Cut Down System

The tracker box also includes the cut down devices (hot wire and tow line). Before attaching the tracker box to a zero pressure balloon, the cut down devices were tested several time in the lab with a HAM radio. These tests were performed to ensure confidence in the cut down system before attaching it to and launching a zero pressure balloon that could be lost without a means to cut it down. A picture of the in-lab testing is shown in Figure 27. The system shown in the figure is upside down for in lab testing. The weight at the bottom of the figure represents the lift force on the system, and the line securing the tracking box from above represents where the sampler box and radar deflector are located.

Figure 27: Tracker Box Testing in Lab

Balloon Flight

Leak checks on zero pressure balloons are performed in the lab before the balloons are used for sampling by filling them with air, as shown Figure 28. After the balloon is filled, a researcher looks, listens, and feels around the balloon for leaks. After initial inspection is complete, the balloon is left inflated for a several minutes and visual inspection of the balloon is done periodically to ensure the absence of major leaking.

An indoor facility was utilized to test the cut down systems attached to a zero pressure balloon without interference from wind. Miller Auditorium at Western Michigan University was the indoor testing facility. Figure 29 shows the filling of a

Figure 28: Leak Checking of ZP Balloon

Figure 29: Filling Zero Pressure Balloon

zero pressure balloon in Miller Auditorium. The balloon was tethered to a spool of string attached to n the load ring to prevent the balloon from going to high. The "Miller Test" consisted of testing both cut down mechanisms and seeing if the parachute would slow the descent of the payload. This tested demonstrated flaws in the tow line release mechanism and showed that the zero pressure balloon deflates much slower than expected. As a result of the Miller Test, modifications to the tow line release mechanism were made, including making a 3D part that would replace the separate pieces of the tow line release and the hotwire, and using wire connecters from the load ring to the parachute to ensure the load ring is level. Furthermore, the size of the load ring was increased to create a bigger hole in the bottom of the balloon from which lift gas can vent. This hole through the load ring

was later replace with the venting duct discussed in the section Manufacturing A Zero Pressure Balloon.

Outdoor testing was a replicate of the Miller test, but with modifications to the tow line mechanism and load ring discussed above. The modifications improved the performance of the tow line release and the rate at which the balloon deflated. During one of the outdoor launched tests, it was visually observed that the parachute slowed the payload descent.

A view of the final attempted launched system can be seen in Figure 30. The cylindrical balloon is at the top of the system, and has the duct extended past the bottom of the balloon. The parachute is connected next, followed by the tracker box. The radar reflector is connected in the middle of the line between the tracker box and the sampler box. Lastly, the sampler box is at the very bottom with the ballast system connected to it.

This test achieved sampling altitude (5 km) in 43 minutes and reached ceiling of 6.4 km in in 62 minutes. It remained above the sampling altitude for 4 hours and 45 minutes, 1 hour and 15 minutes less than the desired time set by the biologists to collect a sample. An Arduino corruption issue occurred 32 minutes into the test, causing the ballast system to start dropping ballast. Figure 31 shows when the ballast was being dropped. There was an increase of 71 ft/min increase in ascent rate from after the ballast stopped dropping till the balloon achieved its ceiling. The ascent rate was 73.7 ft/min, but increased to 144.7 ft/min. This was evidence that

the ballast mechanism worked. Since this flight, the Arduino code has been edited to include a safe guard from corruptions like this happening again.

Figure 30: Launch System After Launching

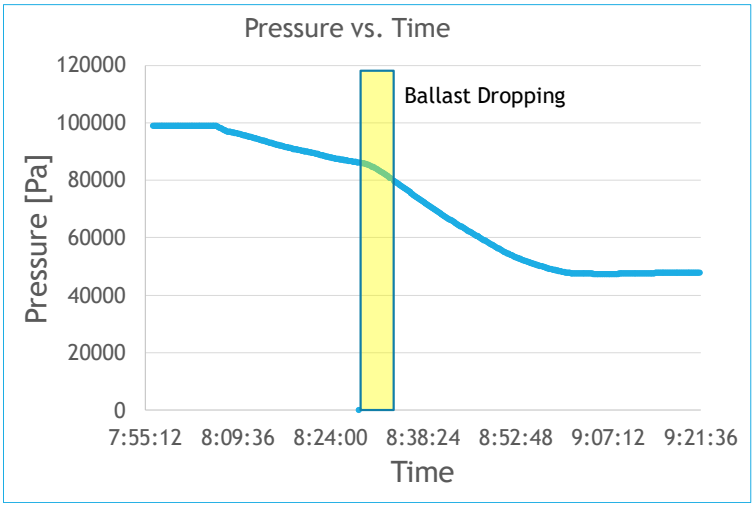

Figure 31: Ballast Dropping with Increased Ascent Rate

Table 5 below summarizes the tests for the launched system.

Table 5: Summary of Tests for the Launched System

Date	Duration	Test	Max Altitude [ft.]
6/10/2016	3 Hours	Latex Balloon - Cut Down Test	83,000
2/15/2017	30 minutes	Miller Auditorium	50
2/27/2017	2 minutes	Outdoor Cut down & Parachute	100
3/2/2017	2 minutes	Outdoor Cut down & Parachute	100
3/3/2017	45 minutes	Latex Balloon - Cut down	9000
3/23/2017	5 minutes	Outdoor Cut down & Parachute	200
4/8/2017	1 Hour 30 minutes	Natural Shape Balloon: Test 1	12,000
4/15/2017	3 Hours	Natural Shape Balloon: Test 2	31,000
6/8/2017	3 Hours	Natural Shape Balloon: Test 3	22,000
6/19/2017	40 minutes	Advertising Balloon	8,000
6/25/2017	1 Hour 30 minutes	Cylindrical Balloon - Test 1	11,000
6/28/2017	4 Hours	Cylindrical Balloon - Test 2	21,000
7/15/2017	4 Hours 45 Minutes	Cylindrical Balloon - Test 3	22,000

CHAPTER VI

RESULTS

Sampler Box Results

To ensure the sampler boxes can collect samples and prevent unwanted samples, a "contamination" experiment was done. The goal of the experiment was to demonstrate that a closed sampler box would result in less extracted DNA and 16S rRNA-based sequences (described below). The results of this experiment demonstrated that the sampler boxes are working as expected, regarding the collection of samples. This experiment used 14 sampler boxes that were all sterilized with ethanol and sterile petri dishes mounted on the G10/FR-4 array. The system was exposed to UV light for 20 minutes. Then 9 sampler boxes were set inside a garage and remained closed. Two sampler boxes were hung in the same garage but were opened to expose petri dishes, and 3 samplers were hung and opened outside of the garage.

The 2 open sampler boxes in the garage, and 3 open boxes outside of the garage were open for 8 hours. The 9 closed sampler boxes stayed in the garage closed for 24 hours. Once this test was complete, the petri plates were taken out of the sampler boxes. Additionally, the box was sterilized with UV lights for 15 minutes before opening the sampler. The petri dishes were swabbed and then the entire swab tip was placed into a bead-beating tube for DNA extraction using a DNEasy PowerWater Kit (Qiagen). These sets of DNA extractions were sent to the

Genomics Core Facility at Michigan State University for 16S rRNA amplicon-based sequencing using an Illumina Mi-Seq approach.

The results received back from Michigan State University after sequencing were evaluated by Kathryn Docherty Ph.D., a principal investigator of the project. The results from the data are summarized in Table 6.

Table 6: Test 16S rRNA-based Sequence Data for Sampler Boxes

	Number of 16S rRNA sequences		
	Closed Box In Garage	Open Box In Garage	Open Box Out of Garage
Sample 1	60	17,600	34,617
Sample 2	73	18,418	37,056
Sample 3	181		39,304
Sample 4	202		
Sample 5	229		
Sample 6	317		
Sample 7	611		
Sample 8	765		
Sample 9	2318		
Average	528.44	18,009	36,992.33
Number of Samples	9	2	3
Standard Deviation	711.85	578.41	2344.15
95% Confidence Interval	465.07	801.63	2652.61

Table 6 shows the number of sequences that were collected from each of the sampler boxes. As shown, the 9 closed sampler boxes had an average number of 528 sequences, with a maximum number of 2318. Although a closed sampler box still

collected unwanted samples, the open sampler boxes collected 2 – 3 orders of magnitude more sequences, as shown in Figure 32.

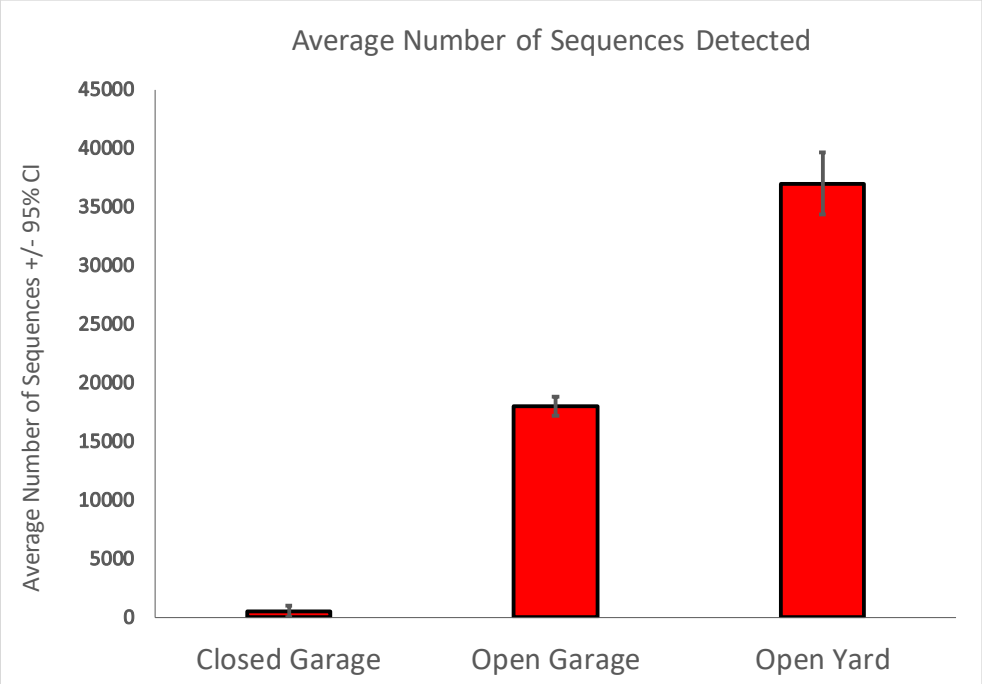

Figure 32: Sequence Data from Contamination Test

This research culminated in a sampling effort in May 2017. Different ecosystems in both rural and urban locations were sampled, and initial results that were analyzed by Dr. Kathryn Docherty also show comparable results regarding the successful sampler box design. This is shown in Figure 33. The data used to develop Figure 33 includes all 82 samples collected from the May 2017 sampling. The figure shows that the boxes used for sampling have on average 3 times more sequences than a sampler box used for a control ("Box Control"). The box control was a sampler box that was sterilized and filled with petri dishes just like the regular sampler boxes. It was brought out into the field but was never opened. Seeing the

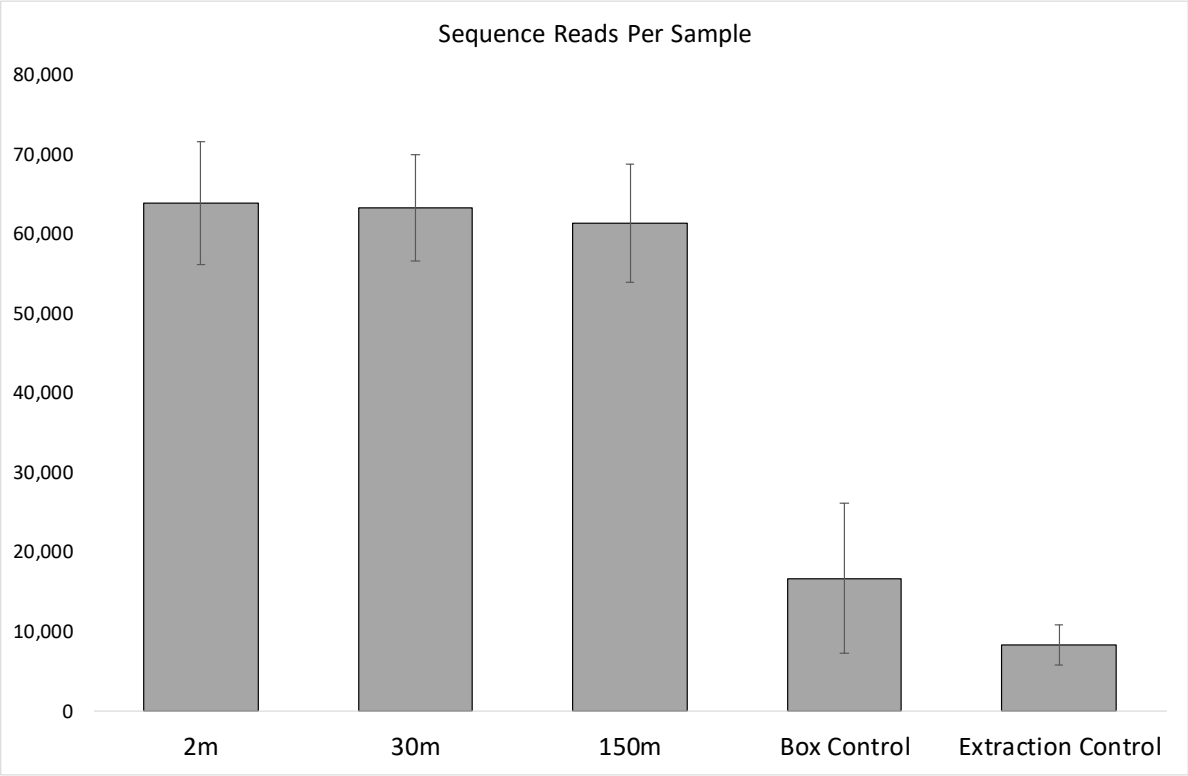

Figure 33: Initial Results of Sampler Boxes from May 2017 Sampling

significant difference in the number of sequences between and open and closed boxes demonstrates that the sampler design is a success. The small level of contaminates there were seen in the controls could have been from several factors: closed cell foam is hard to sterilize, dirt and dust could have been trapped in hard to sterilize sections of the box, etc. None the less, these contaminates can be subtracted from the entire sample population to provide contaminate free results.

Tethered System Results

During the May 2017 sampling effort, the tethered system was used extensively. During this time, the use of advertising balloons allowed the tethered system to maintain sufficient lift to suspend two sampler boxes from the tethering

line of the helikite. In higher wind situations (greater than 15 mph), the tandem advertising balloon was not used. The sampler boxes were all controlled from remotes as mention in the section Sampler Box Functionality. The remotes worked well with minimal issues. Finally, throughout the sampling trip, the equipment, tools, and systems remained operable and safe for the researchers and the public that approached the system.

Launched System Results

The launched system had numerous obstacles that were experienced. Although a zero-pressure balloon showed promising means of being able to perform the desired sampling, the flight duration above 5 km was not long enough. The work that was done here for the launched system could be investigated further and modified to meet the demands that this system requires of it.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

The work done in this thesis provided a robust and flexible system capable of sampling airborne microbial communities at a variety of altitudes. Experiments performed with the sampler box showed that they prevent contamination while closed and have the ability be sterilized. The number of sequences from a test done with closed boxes and open boxes showed that the closed boxes collected almost 3 orders of magnitudes less sequences than open boxes.

The tethered system carried two sampler boxes, one at 30 m and one at 150 m above the ground, for 6 hours. The sampler boxes have remote control capability. that functions over distances greater than 150 m. Helikites were used to ensure stable flight in windy conditions, and tandem balloons were used to sustain altitude in calm conditions. This system was used successfully sampling different ecosystems across the United States in May 2017.

With the complexity of the launched system, the system did not satisfy the requirements of the biologist to collect a sample. Although achieving sampling altitude and maintaining it for a duration of time (4 hours and 45 minutes) was achieved, the sampling time was not long enough to be able to compare the samples between the launched system and the tethered system. The work done by this thesis could lead the way in a solution for future researchers to develop or modify this system that meets the required criteria.

Future Work

Sampler boxes have opportunity for weight optimization. Although the sampler is less than 6 lb, having a lighter sampler box is desired to allow for more lift from both the tethered and launched systems. Optimization of weight on the 3 parts, battery size, and materials could all be considerations in efforts to reduce weight.

While on the May 2017 research effort, numerous times electrical components and 3D had to be repaired or replaced. The abuse the sampler boxes experience in field research can cause these issues and finding ways to improve the ruggedness of the box would also be an improvement for the future work.

Additionally, developing a way or method to prevent insects and pollen from settling on the petri collection dishes would be an improvement desired from the biologist. In the swabbing process of the petri dishes after the sampling has taken place, the biologist swabbing the plates needed to avoid touching insects and other large visible contaminates with the swab. The pollen showed in up in the sequencing results as well, and is not a desirable result to have. Having a form of netting or filtering screening could allow for this undesirable outcome from occurring.

The tethered system requires approximately 2 hours to go from arriving at the sampling site to start sampling. Improving the efficiency of the amount of work

that is required to happen to start sampling would benefit the researchers from performing long days. Furthermore, the tethered system is limited to ground that can/allowed to be staked with long pieces of rebar. Having system that can be used on pavement, fields with irrigations systems, and hard ground would be beneficial to allowing additional sampling locations.

Finally, the launch system requires another test flight to ensure that it is air worthy of collecting samples for the biologist. Once this test is passed, collecting the samples for the biologist would require 3 more flights to collect microbes at 5000 meters. The work done for the launched system could be used for further experiments at different altitudes, and would require a different size balloon.

BIBLIOGRAPHY

[1] N. Yajima, N. Izutsu, T. Imamura and T. Abe, Scientific Ballooning, New York: Springer Science, 2004.

[2] K. Hazlewood, "The Development of Plastic Zero Pressure Balloon Design Since 1945," *Adv. Space Res. ,* vol. 1, pp. 157-161, 1981.

[3] W. V. Jones, "Evolution of scientific balllooning and its impact on astrophysics research," *Advances in Space Research,* vol. 53, no. 10, pp. 1405-1414, 2014.

[4] J. M. Simpson, "Overpressurized Zero Pressure Balloon System," in *International Balloon Technology Conference*, Albuquerque, NM, 1991.

[5] I. S. Smith, "The NASA Balloon Program: looking to the future," *Advances in Space Research,* vol. 33, pp. 1588-1593, 2003.

[6] D. D. Gregory and W. E. Stepp, "NASA's long duration balloon program: the last ten years and the next ten years," *Advances in Space Research,* vol. 33, no. 10, pp. 1608-1612, 2004.

[7] National Aeronutics and Space Administration, "NASA Stratospheric Balloons: Science at the edge of Space," National Science Foundation, 2010.

[8] Q. Dai, D. Xing, X. Fang and Y. Zhao, "Numerical research on the thermal performance of high altitude scientific balloons," *Applied Thermal Engineering,* vol. 114, pp. 51-57, 2017.

[9] R. E. Farley, "BalloonAscent: 3-D Simulation Tool for the Ascent and Float of High-Altitude Balloons," in *AIAA 5th Aviation, Technology, Intefration, and Operations Conference (ATIO)*, Arlington, 2005.

[10] M. K. Heun, R. S. Schlaifer and K. T. Nock, "Trajectory simulation for single balloons and networks," *Advance Space Research,* vol. 30, no. 5, pp. 1239-1244, 2002.

[11] V. V. Kerzhanovich, J. A. Cutts, H. W. Cooper, J. L. Hall, B. A. McDonald, M. T. Pauken, C. V. White, A. H. Yavrouian, A. Castano, H. M. Cathey Jr, D. A. Fairbrother, I. S. Smith, C. M. Shreves, T. Lachenmeier, E. Rainwater and M. Smith, "Breakthrough in Mars balloon technology," *Advances in Space Research,* vol. 33, pp. 1836-1841, 2004.

[12] D. J. Williams, "Thermodynamics and weather balloons," *Weather,* vol. 61, no. 10, pp. 286-287, 2006.

[13] I. S. Smith Jr., "The NASA balloon program: an overview," *Advances in Space Research,* vol. 30, no. 5, pp. 1087-1094, 2002.

[14] N. C. Bryan, M. Stewart, T. G. Guzik and B. C. Christner, "A method for sampling microbial aerosols using high altitude balloons," *Journal of Microbiological Methods,* vol. 107, pp. 161-168, 2014.

[15] M. J. Harris, N. C. Wickramasinghe, D. Lloyd, J. V. Narlikar, P. Rajaratnam, M. P. Turner, S. Al-Mufti, S. Ramadurai and F. Hoyle, "Detection of living cells in stratospheric samples," in *Instruments, Methods, and Missions for Astrobiology IV*, San Diego, CA, 2001.

[16] N. DeLeon-Rodriguez, T. Lathem, L. M. Rodriguez-R, J. M. Barazesh, B. E. Anderson, A. J. Beyersford, L. D. Ziemba, M. Bergin, A. Nenes and K. T. Konstantinidis, "Microbiome of the upper troposphere: Species composition and prevalence, effects of tropical storms, and atmospheric implications," *Proceedings of the National Academy of Sciences of the United States,* vol. 110, no. 7, pp. 2575-2580, 2013.

[17] A. Barberan, J. Ladau, J. W. Leff, K. S. Pollard, H. L. Menninger, R. R. Dunn and N. Fierer, "Continental-scale distributions of dust-associated bacteria and fungi," *Proceedings of the National Academy of Scineces,* vol. 112, no. 18, pp. 5756-5761, 2015.

[18] G. Mhuireach, B. R. Johnson, A. E. Altrichter, J. Ladau, J. F. Meadow, K. S. Pollard and J. L. Green, "Urban greenness influences airborne bacterial community composition," *Scinece of the Total Enviroment,* vol. 571, pp. 680-687, 2016.

[19] N. DeLeon-Rodriguez, T. L. Lathem, L. M. Rodriguea-R, J. M. Barazesh, B. E. Anderson, A. J. Beyersdorf, L. D. Ziemba, M. Bergin, A. Nenes and K. T. Konstantinidis, "Microbiome of the upper troposphere: Species composition and prevalence, effects of tropical storms, and atmospheric implications," *Proceedings of the National Academy of Sciences,* vol. 110, no. 7, pp. 2575-2580, 2012.

[20] D. J. Smith, D. W. Griffin and A. C. Schuerger, "Stratospheric microbiology at 20 km over the Pacific Ocean," *Aerobiologia,* vol. 26, pp. 35-46, 2010.

[21] U.S. Goverment Publishing Office, "Electronic Code of Federal Regualtions," PART 101—MOORED BALLOONS, KITES, AMATEUR ROCKETS, UNMANNED FREE BALLOONS, AND CERTAIN MODEL AIRCRAFT, March 2017. [Online]. Available: http://www.ecfr.gov/cgi-bin/text-idx?rgn=div5&node=14:2.0.1.3.15.

[22] P. AMET, "Zero Pressure Balloon," 17 April 2016. [Online]. Available: http://purdueamet.info/projects/aerospace/guides/zero-pressure-balloon/.

[23] O. C. Winzen, "Modern-day, natural-shape balloon". United States od America Patent 2,526,719, 24 October 1950.

[24] D. D. Gregory, "Implementing new strategic plans for NASA long duration scientific balloons," *Advances in Space Research,* vol. 37, no. 11, pp. 2021-2025, 2006.

[25] National Aeronautics and Space Administration, "Scientific Balloons," Goddard Space Flight Center, [Online]. Available: https://sites.wff.nasa.gov/code820/faq.html. [Accessed 28 June 2017].

[26] NASA, "Type of Balloons," Scientific Balloons, 27 August 2015. [Online]. Available: https://www.nasa.gov/scientific-balloons/types-of-balloons. [Accessed July 2017].

[27] All IFlyTailies, "Tow Releases," IFlyTailies.com, [Online]. Available: https://www.iflytailies.com/store/tow-releases/. [Accessed 21 July 2017].

Appendix A
Arduino Code for Tethered System

```
#include "RTClib.h"
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <Servo.h>
#include <Adafruit_BME280.h>
#define LEVER_SWITCH_PIN 25
int pressSwitch = 0;
int analogPin = 23;
int whilemillis = 0;
int endmillis = 0;

unsigned long samptime = 28800000;

int startTime = 0;
unsigned long previousMillis = 0;
int interval = 10000; //data read and stored once a minute

Servo CrankS, LO1, LO2;

#define hc12 Serial3

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday"};

const int chipSelect = 10;
int nosignal = 52;
int pos = 0;

#define HC12 Serial3

#define ledPin 13         //so we know what blinks
#define BME_VIN 33       //these let us use DIO pins to drive and read the BME280
#define BME_3VO 35      //without extra wires
#define BME_GND 37     //see BME init
#define BME_SCK 39    // Signal CloCK
#define BME_MISO 41  // Master In Slave Out//same as SDO
#define BME_MOSI 43 // Master Out Slave In //same as SDI
#define BME_CS 45  // Cable Select
Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK);//here's where we tell the
library where we are pinned in
bool BME_GO=false;

#define SEALEVELPRESSURE_HPA (1013.25) //hectaPascal
RTC_PCF8523 rtc;

File logfile;

// read a Hex value and return the decimal equivalent
uint8_t parseHex(char c) {
```

```
  if (c < '0')
    return 0;
  if (c <= '9')
    return c - '0';
  if (c < 'A')
    return 0;
  if (c <= 'F')
    return (c - 'A')+10;
}

void error(uint8_t errno) {
  while(1) {
    uint8_t i;
    for (i=0; i<errno; i++) {
      digitalWrite(ledPin, HIGH);
      delay(100);
      digitalWrite(ledPin, LOW);
      delay(100);
    }
    for (i=errno; i<10; i++) {
      delay(200);
    }
  }
}

void setup()
{
 // Open serial communications and wait for port to open:
  Serial.begin(115200);
  //rtc.adjust(DateTime(2016, 10, 16, 11, 31, 0));  //adjust this for the date
  Serial.println(F("BME280 test"));

  //establish BME280 communication
   if (!BME_GO)
       {
         pinMode(BME_GND,OUTPUT);       //This will be ground.
         digitalWrite(BME_GND,LOW);    //We set it to low so it will sink current.
         pinMode(BME_3VO,INPUT);      //We want this pin to float so that we neither load it or drive
it.
         pinMode(BME_VIN,OUTPUT);    //Here's our power source good to 20 mA.
         digitalWrite(BME_VIN,HIGH);//and we turn it on
         pinMode(BME_CS,OUTPUT);   //This is cable select. The library functions drive it but we must
make it output.
         Serial.print("establishing BME280 communication... ");
         if (bme.begin())
         {
           BME_GO=true;
           Serial.println("bme280 GO!");
         }
         else
```

```
      {
        BME_GO=false;
        Serial.println("bme280 NoGO");
        void(*resetfunc)(void) = 0;
        resetfunc();
      }
    }
 while (!Serial) {
  ; // wait for serial port to connect. Needed for Leonardo only
  }

pinMode(6, OUTPUT);
if(Serial.available()) HC12.write(Serial.read());
if(HC12.available()) Serial.write(HC12.read());
digitalWrite(6, LOW); //command mode
HC12.begin(9600);
HC12.print(F("AT+C001\r\n")); //set to channel 1
delay(100);
digitalWrite(6, HIGH);

Serial.print("Initializing SD card...");
// make sure that the default chip select pin is set to
// output, even if you don't use it:
pinMode(chipSelect, OUTPUT);
digitalWrite(chipSelect, HIGH);

// see if the card is present and can be initialized:
if (!SD.begin(chipSelect)) {
  Serial.println("Card failed, or not present");
  // don't do anything more:
  while(1);
}

Serial.println("card initialized.");

char filename[15];
strcpy(filename, "KALLOG00.TXT");
for (uint8_t i = 0; i < 100; i++) {
  filename[6] = '0' + i/10;
  filename[7] = '0' + i%10;
  // create if does not exist, do not open existing, write, sync after write
  if (! SD.exists(filename)) {
    break;
  }
}

logfile = SD.open(filename, FILE_WRITE);
if( ! logfile ) {
  Serial.print("Couldnt create ");
  Serial.println(filename);
```

```
  }
  Serial.print("Writing to ");
  Serial.println(filename);

  Wire.begin();
  if(!rtc.begin()){
    logfile.println("RTC failed");
    Serial.println("Couldn't find RTC");
  }

  logfile.print("Date/time");
  logfile.print(',');
  logfile.print("Altitude (sea level Pressure)");
  logfile.print(',');
  logfile.print("Temperature");
  logfile.print(',');
  logfile.print("Pressure");
  logfile.print(',');
  logfile.print("Humidity");
  logfile.println("\n");
  Serial.print("Ready!");


  pinMode(nosignal, OUTPUT);
  pinMode(LEVER_SWITCH_PIN, INPUT);
  pressSwitch = digitalRead(LEVER_SWITCH_PIN);
  digitalWrite(29, LOW);
  analogWrite(analogPin, 0);
  Serial.println(pressSwitch);
}

void loop()
{
  unsigned long currentMillis = millis();
  DateTime now = rtc.now();

  if (HC12.available()>1){
    int input = HC12.parseInt();

    if (input == 3313) {  //Site 1, 2m box, Down
      //Sampling begins; sampler opens
      Serial.print("Begin Sampling");
      logfile.println("Samplin Begin");

      delay(500);
      logfile.print("Begin Sampling");
      logfile.println();
      logfile.print(now.year(), DEC);
      logfile.print("/");
```

```
  logfile.print(now.month(), DEC);
  logfile.print("/");
  logfile.print(now.day(), DEC);
  logfile.print("/");
  logfile.print(now.hour(), DEC);
  logfile.print(':');
  logfile.print(now.minute(), DEC);
  logfile.print(':');
  logfile.print(now.second(), DEC);
  logfile.print(',');
  logfile.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
  logfile.print(',');
  logfile.print(bme.readTemperature());
  logfile.print(',');
  logfile.print(bme.readPressure());
  logfile.print(',');
  logfile.println(bme.readHumidity());
  logfile.flush();
  delay(1000);


  LO1.attach(4);
  LO2.attach(5);
  delay(500);

      //this line is for the lockout servos
  for (pos = 180; pos >= 0; pos--)
    {
     LO1.write(pos);
     LO2.write(pos);
     delay(5);
      }

  LO1.detach();
  LO2.detach();
  delay(500);
  CrankS.attach(9);
  CrankS.write(0);
  delay(11000);                              // change this value for crank shaft time going down
  CrankS.detach();
  delay(500);
  HC12.print(3379);       //G-A LED flash down
  delay(500);
  HC12.flush();
  void(*resetfunc)(void) = 0;
  resetfunc();
    }

if (input == 3331){ //site 1, Box 2m, Lockout open
  LO1.attach(4);
```

```
   LO2.attach(5);
   delay(500);

  for (pos = 180; pos >= 0; pos--)
    {
     LO1.write(pos);
     LO2.write(pos);
     delay(5);
    }

    LO1.detach();
    LO2.detach();
    delay(500);
    void(*resetfunc)(void) = 0;
     resetfunc();
}

if (input == 3332){ //site 1, Box 2m, lockout close
  LO1.attach(4);
   LO2.attach(5);
   delay(500);

  for (pos = 0; pos <= 180; pos++)
    {
     LO1.write(pos);
     LO2.write(pos);
     delay(5);
    }

    LO1.detach();
    LO2.detach();
    delay(500);
    void(*resetfunc)(void) = 0;
    resetfunc();
}


if (input == 3323 || currentMillis - previousMillis > samptime){ //site 1, box 2m, up(end sample)
  //Sampling ends, box closes

  analogWrite(analogPin, 255);

  Serial.print("back up");
  Serial.println(now.year(), DEC);

  delay(500);
  logfile.print("Sampling End");
  logfile.println();
  logfile.print(now.year(), DEC);
  logfile.print("/");
```

```
logfile.print(now.month(), DEC);
logfile.print("/");
logfile.print(now.day(), DEC);
logfile.print("/");
logfile.print(now.hour(), DEC);
logfile.print(':');
logfile.print(now.minute(), DEC);
logfile.print(':');
logfile.print(now.second(), DEC);
logfile.print(',');
logfile.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
logfile.print(',');
logfile.print(bme.readTemperature());
logfile.print(',');
logfile.print(bme.readPressure());
logfile.print(',');
logfile.println(bme.readHumidity());
logfile.flush();
delay(1000);

Serial.println(pressSwitch);
CrankS.attach(9);
delay(500);

//startcs = millis();
static unsigned long endmillis;
int x = 20000;
endmillis = millis()+ x;

Serial.println(pressSwitch);
millis();
while(pressSwitch==0 &&  millis() < endmillis){
  Serial.println(millis());
  Serial.println(endmillis);
  CrankS.write(180);
  delay(500);
  pressSwitch = digitalRead(LEVER_SWITCH_PIN);
  Serial.println(pressSwitch);

 if(pressSwitch == 1){
    HC12.print(3378); //G-A LED flash up
     delay(500);
    HC12.flush();
  }

}
delay(100);
LO1.attach(4);
LO2.attach(5);
delay(500);
```

```
    for(pos = 0; pos <180; pos++)
      {
      LO1.write(pos);
      LO2.write(pos);
      delay(5);
       }
    LO1.detach();
    LO2.detach();
    CrankS.detach();
    delay(500);

    analogWrite(analogPin, 0);
// }

    void(*resetfunc)(void) = 0;
    resetfunc();
     }
  if (input == 3340){ //site 1, Box 2m, up 1 second
     CrankS.attach(9);
     CrankS.write(180);
     delay(1000);
     CrankS.detach();
     void(*resetfunc)(void) = 0;
      resetfunc();
  }

  if (input == 3350){ //site 1, Box 2m, down 1 second
     CrankS.attach(9);
     CrankS.write(0);
     delay(1000);
     CrankS.detach();
     void(*resetfunc)(void) = 0;
      resetfunc();
  }
 }

 if (currentMillis - previousMillis >= interval){
    Serial.print(now.year(), DEC);
    Serial.print("/");
    Serial.print(now.month(), DEC);
    Serial.print("/");
    Serial.print(now.day(), DEC);
    Serial.print("/");
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.print(',');
```

```
Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
Serial.print(',');
Serial.print(bme.readTemperature());
Serial.print(',');
Serial.print(bme.readPressure());
Serial.print(',');
Serial.println(bme.readHumidity());
delay(500);
logfile.println();
logfile.print(now.year(), DEC);
logfile.print("/");
logfile.print(now.month(), DEC);
logfile.print("/");
logfile.print(now.day(), DEC);
logfile.print("/");
logfile.print(now.hour(), DEC);
logfile.print(':');
logfile.print(now.minute(), DEC);
logfile.print(':');
logfile.print(now.second(), DEC);
logfile.print(',');
logfile.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
logfile.print(',');
logfile.print(bme.readTemperature());
logfile.print(',');
logfile.print(bme.readPressure());
logfile.print(',');
logfile.println(bme.readHumidity());
logfile.flush();
delay(1000);

  previousMillis = currentMillis;
}
 if (currentMillis-startTime >= samptime){ //site 1, box 150m, up(end sample)
//Sampling ends, box closes
Serial.print("back up");
logfile.println("End Sampling");

delay(500);
logfile.print("End Sampling");
logfile.println();
logfile.print(now.year(), DEC);
logfile.print("/");
logfile.print(now.month(), DEC);
logfile.print("/");
logfile.print(now.day(), DEC);
logfile.print("/");
logfile.print(now.hour(), DEC);
logfile.print(':');
logfile.print(now.minute(), DEC);
```

```
logfile.print(':');
logfile.print(now.second(), DEC);
logfile.print(',');
logfile.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
logfile.print(',');
logfile.print(bme.readTemperature());
logfile.print(',');
logfile.print(bme.readPressure());
logfile.print(',');
logfile.println(bme.readHumidity());
logfile.flush();
delay(1000);


//LIMIT SWITCH OPERATION HERE

analogWrite(analogPin, 255);
Serial.println(analogPin);
Serial.println(pressSwitch);
CrankS.attach(9);
delay(500);

while(pressSwitch==0){

  CrankS.write(180);
  delay(500);
  pressSwitch = digitalRead(LEVER_SWITCH_PIN);
  Serial.println(pressSwitch);
}
CrankS.attach(9);
CrankS.write(180);
delay(15000);           //change this for the value for the crank shaft going up

LO1.attach(4);
LO2.attach(5);
delay(500);

for(pos = 0; pos <180; pos++)
  {
  LO1.write(pos);
  LO2.write(pos);
  delay(5);
   }
CrankS.detach();
delay(500);

HC12.print(3378); //G-A LED flash up
HC12.flush();
analogWrite(analogPin, 0);
void(*resetfunc)(void) = 0;
```

```
    resetfunc();
    }
}
```

Appendix B
Arduino Code for Remote of Tethered System

```
#define hc12 Serial3
void setup()
{
  Serial3.begin(9600);
  Serial.begin(250000);
  //A
  pinMode(11, INPUT);
  pinMode(10, INPUT);
  pinMode(13, INPUT);
  pinMode(12, INPUT);
  pinMode(9, INPUT);
  pinMode(8, INPUT);

  //B
  pinMode(5, INPUT);
  pinMode(4, INPUT);
  pinMode(7, INPUT);
  pinMode(6, INPUT);
  pinMode(3, INPUT);
  pinMode(2, INPUT);

  //C
  pinMode(48, INPUT);
  pinMode(46, INPUT);
  pinMode(52, INPUT);
  pinMode(50, INPUT);
  pinMode(44, INPUT);
  pinMode(42, INPUT);

  pinMode(40, OUTPUT);
  pinMode(38, OUTPUT);
  pinMode(36, OUTPUT);

//  pinMode(7,OUTPUT);
//  if(Serial.available()) Serial3.write(Serial.read());
//  if(Serial3.available()) Serial.write(Serial3.read());
//  digitalWrite(7,LOW); // enter AT command mode
//  Serial.begin(9600);
//  Serial3.begin(9600);
//  Serial3.print(F("AT+C001\r\n")); // set to channel 1
//  delay(100);
//  digitalWrite(7,HIGH);// enter transparent mode
}

void loop() {
  //A = 11 10 13 12 9 8
  int Adown = digitalRead(12);
  int Aup = digitalRead(13);
  int ALOO = digitalRead(11);
  int ALOC = digitalRead(10);
```

```
 int Adown2 = digitalRead(8);
 int Aup2 = digitalRead(9);


 //B = 5 4 7 6 3 2
 int Bdown = digitalRead(6);
 int Bup = digitalRead(7);
 int BLOO = digitalRead(5);
 int BLOC = digitalRead(4);
 int Bdown2 = digitalRead(2);
 int Bup2 = digitalRead(3);

 //C = 48 46 52 50 44 42
 int Cdown = digitalRead(50);
 int Cup = digitalRead(52);
 int CLOO = digitalRead(48);
 int CLOC = digitalRead(46);
 int Cdown2 = digitalRead(42);
 int Cup2 = digitalRead(44);

 hc12.flush();

//HC-12 output to tethered box communication
 if(Adown == 1){//A down
   hc12.println(3113);
   Serial.print("\nA down");
 }
 delay(20);//delay little for better serial communication

 if(Aup == 1){//A up
   hc12.println(3123);
   Serial.print("\nA up");
 }
 delay(20);

 if(ALOO == 1){//A lockout open
   hc12.println(3131);
   Serial.print("\nA lockout open");
 }
 delay(20);

 if(ALOC == 1){// A lockout close
   hc12.println(3132);
   Serial.print("\nA lockout close");
 }
 delay(20);

 if(Adown2 == 1){// A down 1 second
   hc12.println(3140);
   Serial.print("\nA down2");
```

```
}
delay(20);//delay little for better serial communication

if(Aup2 == 1){// A up 1 second
  hc12.println(3150);
  Serial.print("\nA up2");
}
delay(20);


if(Bdown == 1){//B down
  hc12.println(3213);
  Serial.print("\nB down");
}
delay(20);

if(Bup == 1){//B up
  hc12.println(3223);
  Serial.print("\nB up");
}
delay(20);

if(BLOO == 1){//B lockout open
  hc12.println(3231);
  Serial.print("\nB lockout open");
}
delay(20);

if(BLOC == 1){//B lockout close
  hc12.println(3232);
  Serial.print("\nB lockout close");
}
delay(20);

if(Bdown2 == 1){// B down 1 second
  hc12.println(3240);
  Serial.print("\nB down2");
}
delay(20);//delay little for better serial communication

if(Bup2 == 1){// B up 1 second
  hc12.println(3250);
  Serial.print("\nB up2");
}
delay(20);


if(Cdown == 1){//C down
  hc12.println(3313);
  Serial.print("\nC down");
```

```
  }
  delay(20);

  if(Cup == 1){//C up
    hc12.println(3323);
    Serial.print("\nC up");
  }
  delay(20);

  if(CLOO == 1){// C lockout open
    hc12.println(3331);
    Serial.print("\nC lockout open");
  }
  delay(20);

  if(CLOC == 1){// C lockout close
    hc12.println(3332);
    Serial.print("\nC lockout close");
  }
  delay(20);

  if(Cdown2 == 1){// C down 1 second
    hc12.println(3340);
    Serial.print("\nC down2");
  }
  delay(20);//delay little for better serial communication

  if(Cup2 == 1){// C up 1 second
    hc12.println(3350);
    Serial.print("\nC up2");
  }
  delay(20);


//HC-12 INPUT communication

 if (hc12.available()>1){
    int input = hc12.parseInt();

    //BOX A LED OPERATION  SLOWER-DOWN     FASTER-UP
    if (input == 3179) {  //Site Green, box A, Down
      digitalWrite(40, HIGH);
      delay(1000);
      digitalWrite(40, LOW);
      delay(500);
      digitalWrite(40, HIGH);
      delay(1000);
      digitalWrite(40, LOW);
      delay(500);
    }
```

```
if (input == 3178){   //Site Green, box A, UP
  digitalWrite(40, HIGH);
  delay(500);
  digitalWrite(40, LOW);
  delay(250);
  digitalWrite(40, HIGH);
  delay(500);
  digitalWrite(40, LOW);
  delay(250);
}

//BOX B LED OPERATION
if (input == 3279) {  //Site Green, box B, Down
  digitalWrite(38, HIGH);
  delay(1000);
  digitalWrite(38, LOW);
  delay(500);
  digitalWrite(38, HIGH);
  delay(1000);
  digitalWrite(38, LOW);
  delay(500);
}

if (input == 3278){   //Site Green, box B, UP
  digitalWrite(38, HIGH);
  delay(500);
  digitalWrite(38, LOW);
  delay(250);
  digitalWrite(38, HIGH);
  delay(500);
  digitalWrite(38, LOW);
  delay(250);
}

//BOX C LED OPERATION
if (input == 3379) {  //Site Green, box C, Down
  digitalWrite(36, HIGH);
  delay(1000);
  digitalWrite(36, LOW);
  delay(500);
  digitalWrite(36, HIGH);
  delay(1000);
  digitalWrite(36, LOW);
  delay(500);
}

if (input == 3378){   //Site Green, box C, UP
  digitalWrite(36, HIGH);
  delay(500);
```

```
      digitalWrite(36, LOW);
      delay(250);
      digitalWrite(36, HIGH);
      delay(500);
      digitalWrite(36, LOW);
      delay(250);
    }
  }

}
```

Appendix C
Arduino Code for Launched Sampler Box

```
#include "RTClib.h"
 #include <stdio.h>         //used for SD file creation
 #include <Adafruit_BME280.h>// barometer, temperature and humidity sensor
 #include <SPI.h>           //required by the BME280 and SD
 #include <SD.h>            // For Logging Data on SD Card
 #include <Servo.h>         // servo support for deployment actuators
 #define SEALEVELPRESSURE_HPA (1013.25) //hectaPascal
 #define HC12 Serial3
 Servo CrankS, LO1, LO2, BAL;
 RTC_PCF8523 rtc;

//limit switch
 #define LEVER_SWITCH_PIN 25
 int pressSwitch = 0;
 int reedSwitch = 0;
 int analogPin = 23;
 int whilemillis = 0;
 int endmillis = 0;
 const int REED_PIN = 2;
 #define ledPin 13         //so we know what blinks
 #define BME_VIN 33        //these let us use DIO pins to drive and read the BME280
 #define BME_3VO 35        //without extra wires
 #define BME_GND 37        //see BME init
 #define BME_SCK 39    // Signal CloCK
 #define BME_MISO 41  // Master In Slave Out//same as SDO
 #define BME_MOSI 43 // Master Out Slave In //same as SDI
 #define BME_CS 45  // Cable Select
 Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK);//here's where we tell the
library where we are pinned in
 bool BME_GO=false;

//Creation of case structure. If you wish to add an additional case, just simple add the name here
 enum programState {ascend, deploy, collect, reploy, descend} state;

//////////////////////////Variables, timers, and pin select/////////////////////////////////
 const int chipSelect = 10;             //SD data
 unsigned long startTime = 0;                    //beginning of sample time
 int pos = 0;                           //lockout servos
 int posbal = 0;                        // initial position of the ballast servo
 int record = 10000;                    //Records data every ten seconds
 unsigned long sampTime = 7200000;    //Sample time - 2 hours. (millis)
 unsigned long previousMillis = 0;
 unsigned long PressureAltitude = 54019;          //Pressure in Pa at 5km
 unsigned long openanyway = 3600000;    //Open after 1 hour in air (millis)**** 1 hour = 3600000 ms
 unsigned long halfhour = 1800000;      //half hour after box opens
 unsigned long pbottom = 4500;         //if pressure varies by more than this, in pascals, do something
**** = 4500
//  unsigned long pbottom2 = 57728;       //pressure at 4.5km   **** = 57728 Pa
 int balcount = 0;
 unsigned long CurrentPressure = 100000000;    // Current Pressure
```

```
  File logfile;

  // read a Hex value and return the decimal equivalent
  uint8_t parseHex(char c) {
    if (c < '0')
      return 0;
    if (c <= '9')
      return c - '0';
    if (c < 'A')
      return 0;
    if (c <= 'F')
      return (c - 'A')+10;
  }

  void error(uint8_t errno) {
    while(1) {
      uint8_t i;
      for (i=0; i<errno; i++) {
        digitalWrite(ledPin, HIGH);
        delay(100);
        digitalWrite(ledPin, LOW);
        delay(100);
      }
      for (i=errno; i<10; i++) {
        delay(200);
      }
    }
  }

void setup() {

  Serial.begin(250000);
  Serial.println("\r\nBalloon Data Logger: Launched code");

  //BME Setup
  //establish BME280 communication
   if (!BME_GO)
        {
          pinMode(BME_GND,OUTPUT);       //This will be ground.
          digitalWrite(BME_GND,LOW);    //We set it to low so it will sink current.
          pinMode(BME_3VO,INPUT);      //We want this pin to float so that we neither load it or drive
it.
          pinMode(BME_VIN,OUTPUT);    //Here's our power source good to 20 mA.
          digitalWrite(BME_VIN,HIGH);//and we turn it on
          pinMode(BME_CS,OUTPUT);   //This is cable select. The library functions drive it but we must
make it output.
          Serial.print("establishing BME280 communication... ");
          if (bme.begin())
          {
```

```
        BME_GO=true;
        Serial.println("bme280 GO!");
      }
      else
      {
        BME_GO=false;
        Serial.println("bme280 NoGO");
        void(*resetfunc)(void) = 0;
        resetfunc();
      }
    }
//HC12 setup
  pinMode(6, OUTPUT);
  if(Serial.available()) HC12.write(Serial.read());
  if(HC12.available()) Serial.write(HC12.read());
  digitalWrite(6, LOW); //command mode
  HC12.begin(9600);
  HC12.print(F("AT+C001\r\n")); //set to channel 1
  delay(100);
  digitalWrite(6, HIGH);

//SD setup
  Serial.print("Initializing SD card...");
    pinMode(chipSelect, OUTPUT);
    digitalWrite(chipSelect, HIGH);
// see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    // don't do anything more:
    while(1);
  }

  Serial.println("card initialized.");

  //Creation of the file
  char filename[15];
  strcpy(filename, "KALLOG00.TXT");
  for (uint8_t i = 0; i < 100; i++) {
    filename[6] = '0' + i/10;
    filename[7] = '0' + i%10;
    // create if does not exist, do not open existing, write, sync after write
    if (! SD.exists(filename)) {
      break;
    }
  }

  logfile = SD.open(filename, FILE_WRITE);
  if( ! logfile ) {
    Serial.print("Couldnt create ");
    Serial.println(filename);
```

```
  }
  Serial.print("Writing to ");
  Serial.println(filename);

//initializing RTC clock
  Wire.begin();
  if(!rtc.begin()){
    logfile.println("RTC failed");
    Serial.println("Couldn't find RTC");
  }

//tops of the columns for the text file
  logfile.print("Date/time");
  logfile.print(',');
  logfile.print("Altitude (sea level Pressure)");
  logfile.print(',');
  logfile.print("Temperature");
  logfile.print(',');
  logfile.print("Pressure");
  logfile.print(',');
  logfile.print("Humidity");
  logfile.println("\n");
  Serial.print("Ready!");

  //Limit switch initialization
  pinMode(LEVER_SWITCH_PIN, INPUT);
  pinMode(REED_PIN, INPUT_PULLUP);
  pressSwitch = digitalRead(LEVER_SWITCH_PIN);
  reedSwitch = digitalRead(REED_PIN);
  digitalWrite(29, LOW);
  analogWrite(analogPin, 0);
  Serial.println(pressSwitch);
}

 void loop() {
  unsigned long currentMillis = millis();
  DateTime now = rtc.now();
//////////////////////////////Begin Case Structure////////////////////////////////
  switch (state)
  {
   case ascend:

    CurrentPressure = (bme.readPressure());

    if (currentMillis - previousMillis >= record){
      collector();
      previousMillis = currentMillis;

    }
```

```
 if(currentMillis>=openanyway){
   Serial.println("Box opened due to 1 hour in air reached");
   logfile.println("Box opened due to 1 hour in air reached");
   delay(500);
   PressureAltitude = 61000;
   logfile.print("pressure for sampling is ");
   logfile.println(CurrentPressure);
   Serial.print("pressure for sampling is ");
   Serial.println(CurrentPressure);

   state=deploy;  //run deploy case if the 1 hour in air
 }

 if(CurrentPressure <= PressureAltitude){
   Serial.println(CurrentPressure);
   logfile.print("Current Pressure = ");
   logfile.println(CurrentPressure);
   Serial.println("5km reached. Yey!");
   logfile.println("5km reached. Yey!");
   delay(500);
   state=deploy;  //run deploy at 5km
 }

 if(HC12.available()>1){
     int input = HC12.parseInt();
     Serial.println("GOT IT");
     logfile.println("GOT IT");
     Serial.println("Recieved Signal from Tracker Box");
     logfile.println("Recieved Signal from Tracker Box");
     Serial.println(input);

    // HC12 Commands From Mini
   // * & 1 is 1424 (AKA HotWire Command)
       if(input == 1424){
         Serial.println("Burn Baby Burn! Hot Wire Command Triggered");
         hotwire();
       }
       if (input == 3863){
         Serial.println("LET IT GO! Tow Line Release Command Triggered");
         releaseTowline();
         delay(500);
       }

   }


break;

case deploy:
```

```
      Serial.println("Sampler Open Started");
      logfile.println("Sampler Open Started");
      deployTray();

      //start timing sampling
      startTime = currentMillis;
      Serial.println("Sampler is Open");
      logfile.println("Sampler is Open");
      state=collect;
  break;

  case collect:
//    if(currentMillis-startTime<=halfhour){
//      if (currentMillis - previousMillis >= record){
//        collector();
//        previousMillis = currentMillis;
//      }
//      if(currentMillis-startTime>=halfhour && bme.readPressure()>=pbottom){
//        Serial.println("Box closing. Going below 4km");
//        logfile.println("Box closing. Going below 4km");
//        logfile.println();
//        state=reploy;
//      }
//    }
//    if(currentMillis - startTime>=halfhour){

//record weather data
     if (currentMillis - previousMillis >= record){
       collector();
       previousMillis = currentMillis;

     }

// if pressure is 4500 Pa more than the pressure at the designated altitude, then the balloon is too
low, drop ballast, as long as there is still ballast to drop
     if(bme.readPressure()>=(PressureAltitude+pbottom) && balcount<77){
       Serial.println("Dropping Ballast. Going below 4.5km");
       logfile.println("Dropping Ballast. Going below 4.5km");
       logfile.print("Ballast cycle number");
       logfile.println(balcount);
       Serial.println(bme.readPressure());
       Serial.println(balcount);
       Ballast();
       logfile.println();
     }

// if pressure is 9000 Pa more than the pressure at the designated altitude, then the balloon is WAY
too low, end sampling
       if(bme.readPressure()>=(PressureAltitude+2*pbottom)  && balcount >= 77 ){
       Serial.println("Closing Box. Going below 4 km");
```

```
    logfile.println("Closing Box. Going below 4 km");
    logfile.println();
    state=reploy;
   }
   if(currentMillis-startTime>=sampTime){
    Serial.println("Box cloxing. Time reached");
    logfile.println("Box closing. Time reached");
    logfile.println();
    state=reploy;
   }

 if(HC12.available()>1){
    int input = HC12.parseInt();
    Serial.println("GOT IT");
    logfile.println("GOT IT");
    Serial.println("Recieved Signal from Tracker Box");
    logfile.println("Recieved Signal from Tracker Box");
    Serial.println(input);

    // HC12 Commands From Mini
   // * & 1 is 1424 (AKA HotWire Command)
      if(input == 1424){
       Serial.println("Burn Baby Burn! Hot Wire Command Triggered");
       hotwire();
      }
      if (input == 3863){
       Serial.println("Tow Line Release Command Triggered");
       releaseTowline();
       delay(500);
      }
   }
 break;

 case reploy:
    Serial.println("Sampler Close Started");
    logfile.println("Sampler Close Started");
    reployTray();
    Serial.println("Sampler is Closed");
    logfile.println("Sampler is Closed");
    state=descend;
 break;

case descend:
  if (currentMillis - previousMillis >= record){
   collector();
   previousMillis = currentMillis;
  }

   if(HC12.available()>1){
      int input = HC12.parseInt();
```

```
        Serial.println("GOT IT");
        logfile.println("GOT IT");
        Serial.println("Recieved Signal from Tracker Box");
        logfile.println("Recieved Signal from Tracker Box");
        Serial.println(input);

       // HC12 Commands From Mini
      // * & 1 is 1424 (AKA HotWire Command)
         if(input == 1424){
           Serial.println("Burn Baby Burn! Hot Wire Command Triggered");
           logfile.println("Burn Baby Burn! Hot Wire Command Triggered");
           hotwire();
           hotwire();
         }
         if (input == 3863){
           Serial.println("Tow Line Release Command Triggered");
           logfile.println("Tow Line Release Command Triggered");
           releaseTowline();
           delay(500);
         }
       }
  break;
  }
}

//This is the collector method. It runs fairly similar to an interrupt. It runs in all cases except for
reploy and deploy
void collector(){
  DateTime now = rtc.now();
     Serial.print(now.year(), DEC);
     Serial.print("/");
     Serial.print(now.month(), DEC);
     Serial.print("/");
     Serial.print(now.day(), DEC);
     Serial.print("/");
     Serial.print(now.hour(), DEC);
     Serial.print(':');
     Serial.print(now.minute(), DEC);
     Serial.print(':');
     Serial.print(now.second(), DEC);
     Serial.print(',');
     Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
     Serial.print(',');
     Serial.print(bme.readTemperature());
     Serial.print(',');
     Serial.print(bme.readPressure());
     Serial.print(',');
     Serial.println(bme.readHumidity());
     delay(500);
     logfile.println();
```

```
    logfile.print(now.year(), DEC);
    logfile.print("/");
    logfile.print(now.month(), DEC);
    logfile.print("/");
    logfile.print(now.day(), DEC);
    logfile.print("/");
    logfile.print(now.hour(), DEC);
    logfile.print(':');
    logfile.print(now.minute(), DEC);
    logfile.print(':');
    logfile.print(now.second(), DEC);
    logfile.print(',');
    logfile.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
    logfile.print(',');
    logfile.print(bme.readTemperature());
    logfile.print(',');
    logfile.print(bme.readPressure());
    logfile.print(',');
    logfile.println(bme.readHumidity());
    logfile.flush();
    delay(500);

}

//deploy trays method
void deployTray()
{
    Serial.println("Begin Sampling");
    logfile.println("Samplin Begin");
    DateTime now = rtc.now();

    delay(500);
    logfile.print("Begin Sampling");
    logfile.println();
    logfile.print(now.year(), DEC);
    logfile.print("/");
    logfile.print(now.month(), DEC);
    logfile.print("/");
    logfile.print(now.day(), DEC);
    logfile.print("/");
    logfile.print(now.hour(), DEC);
    logfile.print(':');
    logfile.print(now.minute(), DEC);
    logfile.print(':');
    logfile.print(now.second(), DEC);
    logfile.print(',');
    logfile.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
    logfile.print(',');
    logfile.print(bme.readTemperature());
    logfile.print(',');
```

```
    logfile.print(bme.readPressure());
    logfile.print(',');
    logfile.println(bme.readHumidity());
    logfile.flush();
    delay(1000);


    LO1.attach(4);
    LO2.attach(5);
    delay(500);

        //this line is for the lockout servos
    for (pos = 180; pos >= 0; pos--)
      {
       LO1.write(pos);
       LO2.write(pos);
       delay(5);
        }

    LO1.detach();
    LO2.detach();
    delay(500);
    CrankS.attach(9);

    static unsigned long endmillis_reed;
    int y = 11000;
    endmillis_reed = millis()+ y;
// REED SWITCH

  while(reedSwitch==1 &&  millis() < endmillis_reed){
      CrankS.write(0);
      delay(500);
      reedSwitch = digitalRead(REED_PIN);
      }
   // delay(11000); // change this value for crank shaft time going down
    CrankS.detach();
    HC12.println(9999); //tells tracker box that the sampler is open
    HC12.flush();
    return;
}

void reployTray(){
    DateTime now = rtc.now();

    analogWrite(analogPin, 255);
    Serial.println(now.year(), DEC);

    delay(500);
    logfile.print("Sampling End");
    logfile.println();
```

```
logfile.print(now.year(), DEC);
logfile.print("/");
logfile.print(now.month(), DEC);
logfile.print("/");
logfile.print(now.day(), DEC);
logfile.print("/");
logfile.print(now.hour(), DEC);
logfile.print(':');
logfile.print(now.minute(), DEC);
logfile.print(':');
logfile.print(now.second(), DEC);
logfile.print(',');
logfile.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
logfile.print(',');
logfile.print(bme.readTemperature());
logfile.print(',');
logfile.print(bme.readPressure());
logfile.print(',');
logfile.println(bme.readHumidity());
logfile.flush();
delay(1000);

Serial.println(pressSwitch);
CrankS.attach(9);
delay(500);

//startcs = millis();
static unsigned long endmillis;
int x = 20000;
endmillis = millis()+ x;

Serial.println(pressSwitch);
millis();
while(pressSwitch==0 &&  millis() < endmillis){
  Serial.println(millis());
  Serial.println(endmillis);
  CrankS.write(180);
  delay(500);
  pressSwitch = digitalRead(LEVER_SWITCH_PIN);
  Serial.println(pressSwitch);

}
delay(100);
LO1.attach(4);
LO2.attach(5);
delay(500);

for(pos = 0; pos <180; pos++)
  {
  LO1.write(pos);
```

```
      LO2.write(pos);
      delay(5);
       }
    LO1.detach();
    LO2.detach();
    CrankS.detach();
    delay(500);
    analogWrite(analogPin, 0);

    HC12.println(1111);
    logfile.println("Hot Wire Command Sent To Tracker Box");
    delay(15000);
    HC12.println(2222);
    logfile.println("Release Line Command Sent To Tracker Box");
    return;
}

void hotwire(){
  Serial.println("\n\nHOT WIRE");
  logfile.println("\n\nHOT WIRE");

  DateTime now = rtc.now();

    analogWrite(analogPin, 255);
    Serial.println(now.year(), DEC);

    delay(500);
    logfile.print("Sampling End");
    logfile.println();
    logfile.print(now.year(), DEC);
    logfile.print("/");
    logfile.print(now.month(), DEC);
    logfile.print("/");
    logfile.print(now.day(), DEC);
    logfile.print("/");
    logfile.print(now.hour(), DEC);
    logfile.print(':');
    logfile.print(now.minute(), DEC);
    logfile.print(':');
    logfile.print(now.second(), DEC);
    logfile.print(',');
    logfile.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
    logfile.print(',');
    logfile.print(bme.readTemperature());
    logfile.print(',');
    logfile.print(bme.readPressure());
    logfile.print(',');
    logfile.println(bme.readHumidity());
    logfile.flush();
    delay(1000);
```

```
    Serial.println(pressSwitch);
    CrankS.attach(9);
    delay(500);

    //startcs = millis();
    static unsigned long endmillis;
    int x = 20000;
    endmillis = millis()+ x;

    Serial.println(pressSwitch);
    millis();
    while(pressSwitch==0 &&  millis() < endmillis){
      Serial.println(millis());
      Serial.println(endmillis);
      CrankS.write(180);
      delay(500);
      pressSwitch = digitalRead(LEVER_SWITCH_PIN);
      Serial.println(pressSwitch);

    }
    delay(100);
    LO1.attach(4);
    LO2.attach(5);
    delay(500);

    for(pos = 0; pos <180; pos++)
      {
      LO1.write(pos);
      LO2.write(pos);
      delay(5);
        }
    LO1.detach();
    LO2.detach();
    CrankS.detach();
    delay(500);
    analogWrite(analogPin, 0);
  Serial.println("Sampling Complete");
  HC12.flush();
  delay(1000);
  HC12.print(1111);
  return;
}

void Ballast(){
BAL.attach(10);
      delay(500);
Serial.println("entering ballast");

    //open ballast
```

```
    for(posbal = 150; posbal >75; posbal--)
     {
     BAL.write(posbal);
     delay(5);
     }

    for(posbal = 75; posbal <150; posbal++)
     {
     BAL.write(posbal);
     delay(5);
     }
    BAL.detach();
    delay(2000);

    Serial.println("Dropping Ballast ended");
    logfile.println("Dropping Ballast ended");
    balcount = balcount+1;
}


void releaseTowline(){
  Serial.println("\n\nRelease Tow Line");
  logfile.println("\n\nRelease Tow Line");
  DateTime now = rtc.now();

    analogWrite(analogPin, 255);
    Serial.println(now.year(), DEC);

    delay(500);
    logfile.print("Sampling End");
    logfile.println();
    logfile.print(now.year(), DEC);
    logfile.print("/");
    logfile.print(now.month(), DEC);
    logfile.print("/");
    logfile.print(now.day(), DEC);
    logfile.print("/");
    logfile.print(now.hour(), DEC);
    logfile.print(':');
    logfile.print(now.minute(), DEC);
    logfile.print(':');
    logfile.print(now.second(), DEC);
    logfile.print(',');
    logfile.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
    logfile.print(',');
    logfile.print(bme.readTemperature());
    logfile.print(',');
    logfile.print(bme.readPressure());
    logfile.print(',');
```

```
    logfile.println(bme.readHumidity());
    logfile.flush();
    delay(1000);

    Serial.println(pressSwitch);
    CrankS.attach(9);
    delay(500);

    //startcs = millis();
    static unsigned long endmillis;
    int x = 20000;
    endmillis = millis()+ x;

    Serial.println(pressSwitch);
    millis();
    while(pressSwitch==0 &&  millis() < endmillis){
      Serial.println(millis());
      Serial.println(endmillis);
      CrankS.write(180);
      delay(500);
      pressSwitch = digitalRead(LEVER_SWITCH_PIN);
      Serial.println(pressSwitch);

    }
    delay(100);
    LO1.attach(4);
    LO2.attach(5);
    delay(500);

    for(pos = 0; pos <180; pos++)
      {
      LO1.write(pos);
      LO2.write(pos);
      delay(5);
       }
    LO1.detach();
    LO2.detach();
    CrankS.detach();
    delay(500);
    analogWrite(analogPin, 0);
  Serial.println("Sampling Complete");
  Serial.println("REPLOY RAN");
  Serial.println("2222");
  HC12.println(2222); //tow line
  return;
}
```

Appendix D
Arduino Code for Launched Tracker Bo

```
#include <Servo.h>
  int n = 0; //hotwire
  int w = 0; //motor
  int f =0; //mexicao emergency
  int k = 0;
  int p=0;
  int pos = 0;
  int startTime=0;
  int currentmillis=0;
  unsigned long timer1 = 14400000; //4 hours
  unsigned long timer2 = 12600000; //3.5 hours

  #define HC12 Serial
  Servo servo;

// Void Set Up
  void setup(){
    Serial.begin(9600);
    servo.attach(10);

// DTMF Inputs

    pinMode(2, INPUT); // 4 on DTMF
    pinMode(13, INPUT); // 7 on DTMF
    pinMode(12, INPUT); // 6 on DTMF
    pinMode(11, INPUT); // "5"
    pinMode(9, OUTPUT); //LED flash
    pinMode(8, INPUT); //*
    pinMode(5, INPUT); //1
    pinMode(6, OUTPUT); //hot wire
    pinMode(7, INPUT); //#
    pinMode(4, INPUT);  //2
    pinMode(10, OUTPUT); //motor

    for (pos = 0; pos <= 90; pos ++){
        servo.write(pos);
        delay(15);
     }
     servo.detach();
    }

// Void Loop
void loop(){
// Inputs from Sampler Box HC 12
  currentmillis = millis();


  if(HC12.available()>1){
    int input = HC12.parseInt();
      Serial.println(input);
```

```
    if(input == 1111){
      HotWireCutLine();
    }
    if(input == 2222){
      ReleaseTowLine();
    }
    if(input == 9999){
      p++;
      unsigned long openbox = currentmillis;
    }
  }

// LED Blink Light Code
  if(digitalRead(11) == HIGH){ //LED
    Serial.print("5555");
    digitalWrite(9, HIGH);
    delay(500);
    digitalWrite(9, LOW);
    delay(250);
    digitalWrite(9, HIGH);
    delay(500);
    digitalWrite(9, LOW);
    delay(250);
    digitalWrite(9, HIGH);
    delay(500);
    digitalWrite(9, LOW);
    delay(250);
    digitalWrite(9, HIGH);
    delay(500);
    digitalWrite(9, LOW);
    delay(250);
    digitalWrite(9, HIGH);
    delay(500);
    digitalWrite(9, LOW);
    delay(250);
  }


// DTMF: PTT + * + 1 Code
  if(digitalRead(8) == HIGH){
    n++;
  }

  if((digitalRead(5) == HIGH) && n >=1){

    digitalWrite(9, HIGH);
    delay(500);
    digitalWrite(9, LOW);
    delay(250);
    digitalWrite(9, HIGH);
```

```
    delay(500);
    digitalWrite(9, LOW);
    delay(250);

      Serial.println(1424);

      n=0;
      Serial.flush();
  }

// DTMF: PTT + # + 2 Code
  if(digitalRead(7) == HIGH){
    w++;
  }
  if((digitalRead(4) == HIGH) && w>=1){
    digitalWrite(9, HIGH);
    delay(500);
    digitalWrite(9, LOW);
    delay(250);
    digitalWrite(9, HIGH);
    delay(500);
    digitalWrite(9, LOW);
    delay(250);
    //Tell the mega code to close the box

      Serial.println(3863);
      delay(500);
    w=0;
    delay(500);
  }


// Mexico Cut Down Emergency Operation
  // DTMF = 7 + 4 + 6
  if(digitalRead(13) == HIGH){
    f++;
  }
  if(digitalRead(2) == HIGH && f>=1){
    k++;
  }
  if(digitalRead(12) == HIGH && k>=1){ //Mexico Emergency
    HotWireCutLine();
    delay(500);
    ReleaseTowLine();
    delay(500);
    Serial.println(6336); //Told mega emergency cut down
    Serial.println("Mexico Emergency triggered by DTMF");
    f=0;
    w=0;
    n=0;
```

```
    k=0;
  }

//
//  if(currentmillis >= timer1 && p==0){
//    ReleaseTowLine();
//    delay(1000);
//    HotWireCutLine();
//  }
//
//  if(currentmillis>=timer2 && p>=1){
//    ReleaseTowLine();
//    delay(1000);
//    HotWireCutLine();
//  }
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Methods called from loop

// Tow Line Release Function
void ReleaseTowLine(){
  servo.attach(10);
    for(pos = 90; pos >=0; pos--){
      servo.write(pos);
      delay(15);
    }
    servo.detach();
    return;
  }
// Hot Wire Function
void HotWireCutLine() {
    digitalWrite(6, HIGH);
    delay(10000);
    digitalWrite(6, LOW);
    return;
    }
```

Appendix E
Balloon Template Values

At 0.0 cm from bottom, gore is 0.00 cm from center
At 10.0 cm from bottom, gore is 3.07 cm from center
At 20.0 cm from bottom, gore is 6.13 cm from center
At 30.0 cm from bottom, gore is 9.19 cm from center
At 40.0 cm from bottom, gore is 12.25 cm from center
At 50.0 cm from bottom, gore is 15.29 cm from center
At 60.0 cm from bottom, gore is 18.30 cm from center
At 70.0 cm from bottom, gore is 21.28 cm from center
At 80.0 cm from bottom, gore is 24.23 cm from center
At 90.0 cm from bottom, gore is 27.10 cm from center
At 100.0 cm from bottom, gore is 29.91 cm from center
At 110.0 cm from bottom, gore is 32.61 cm from center
At 120.0 cm from bottom, gore is 35.19 cm from center
At 130.0 cm from bottom, gore is 37.63 cm from center
At 140.0 cm from bottom, gore is 39.88 cm from center
At 150.0 cm from bottom, gore is 41.90 cm from center
At 160.0 cm from bottom, gore is 43.68 cm from center
At 170.0 cm from bottom, gore is 45.13 cm from center
At 180.0 cm from bottom, gore is 46.23 cm from center
At 190.0 cm from bottom, gore is 46.95 cm from center
At 200.0 cm from bottom, gore is 47.19 cm from center
At 210.0 cm from bottom, gore is 46.98 cm from center
At 220.0 cm from bottom, gore is 46.26 cm from center
At 230.0 cm from bottom, gore is 45.02 cm from center
At 240.0 cm from bottom, gore is 43.31 cm from center
At 250.0 cm from bottom, gore is 41.10 cm from center
At 260.0 cm from bottom, gore is 38.48 cm from center
At 270.0 cm from bottom, gore is 35.52 cm from center
At 280.0 cm from bottom, gore is 32.23 cm from center
At 290.0 cm from bottom, gore is 28.73 cm from center
At 300.0 cm from bottom, gore is 25.06 cm from center
At 310.0 cm from bottom, gore is 21.27 cm from center
At 320.0 cm from bottom, gore is 17.41 cm from center
At 330.0 cm from bottom, gore is 13.51 cm from center
At 340.0 cm from bottom, gore is 9.59 cm from center
At 350.0 cm from bottom, gore is 5.67 cm from center
At 360.0 cm from bottom, gore is 1.74 cm from center

Appendix F
Balloon Lift Calculations MATLAB Code

```
%% Code for Ballooning
    % Written by: Kenneth Domingue
    % Note: SI Units

    close all
    clear all
    clc
%% Constants and Defining Altitude
        alt = 0:8:3200;                     % Altitude [m]
        grvty   =   9.81;                   % Gravity [m/s^2]
        Ntolb   =   0.224809;               % Conversion from [N] to [lbf]
        kgtolb  =   2.20462;                % Conversion from [kg] to [lb]
        lbtoN   =   4.448;                  % Conversion from [lb] to [N]
        TEA     =   11.93                   % Weight of the TEA [lb]
        Poles   =   3.77                    % Weight of the TEA Poles [lb]
        SPOT    =   3.6/16                  % Weight of the SPOT Trace [lb]
        LandS   =   6.5/16                  % Weight of the Streamers and
Lights [lb]
        Teth    =   5                       % Approx Guess Weight of Tether
[lb] (525 feet *1/128 pounds per foot)
        Samplers=   5*2                     % Approx Weight of the Samplers
[lb]
        W       =   TEA+Poles+SPOT+LandS+Teth+Samplers  % Weight of system
[lb]
        W_N     =   W*lbtoN;                    % Weight of system [N]
%% Altitude Loop
    for j = 1:length(alt)
        D       =   tsa(0:8:3200,'si');     % Dummy Index for TSA
        p       =   D(:,1);                  % Pressure at alt [kg/m^2]
        t       =   D(:,2);                  % Temperature at alt [Kelvin]
        rho     =   D(:,3);                  % Air Density at alt[kg/m^3]
        sigma   =   D(:,4);                  % Air Density at alt/ Air Density
at SL [-]
    end
%% Helium Calculations

    R_He        = 2077;                      % Individual Gas Constant for
Helium [J/kg*K]
    R_dry_air   = 287.05;                    % Individual Gas Constant for Dry
Air[J/kg*K]
    R_wet_air   = 461.5;                     % Individual Gas Constant for Wet
Air[J/kg*K]
    Pc          = 16547000                  % Pressure of cyclinder of helium
[Pa]
    Vc          = 0.04899732                    % Volume of 300 Cyclinder
[m^3]
    rho_He_C    = Pc/(R_He*294.261)         % Density of Helium in Cyclinder
[kg/m^3]
    m_He_C      = rho_He_C*Vc               % Mass of He per cylinder [kg]

%% Code Benchmarking - Need 15m^3 for Helikite

    V_h     = 16        % Volume of full helikite [m^3]
```

```
% Kalamazoo December 23rd, 2017

    P_kzoo_1223 = 99016.38            % Sea Level Pressure on 12/23/2016 in
Kalamazoo from Arduino Data [Pa]
    T_kzoo_1223 = 275.372             % Average Temperature on 12/23/2016 in
Kalamazoo from Arduino Data [K]
    Pws_kzoo_1223 = (2.718^(77.3450+0.0057*(T_kzoo_1223)...
                        -(7235/T_kzoo_1223)))/(T_kzoo_1223^8.2);
    x_kzoo_1223 = 0.62198*(Pws_kzoo_1223)/(P_kzoo_1223-Pws_kzoo_1223);
    rho_kzoo_1223 = ((P_kzoo_1223/(R_dry_air*T_kzoo_1223))*...
            (1+x_kzoo_1223))/(1+(x_kzoo_1223*(R_wet_air/R_dry_air)));    %
Air Density on 12/23/2016 in Kalamazoo [kg/m^3]
    n_kzoo_1223 = (P_kzoo_1223*V_h)/(m_He_C*R_He*T_kzoo_1223)   % Number of
Cylinders needed [cyclinders]
    rho_he_kzoo_1223 = P_kzoo_1223/((R_He*T_kzoo_1223));         % Helium
Density in Helikite on 12/23/2016 in Kalamazoo [kg/m^3]
    Fb_kzoo_1223 = ((rho_kzoo_1223)*grvty*V_h)*Ntolb % Force from boyancy
force from helium [lbf]
    Fb_kzoo_1223_N  = ((rho_kzoo_1223)*grvty*V_h); % Force from boyancy force
from helium [lbf]
    ExcessL_kzoo_1223 = Fb_kzoo_1223 - W - m_He_C*n_kzoo_1223*Ntolb;
    ExcessL_N_kzoo_1223 = Fb_kzoo_1223_N - W_N- m_He_C*n_kzoo_1223

% Pellston January 13th, 2017
    P_pell_113 = 101271.69           % Air Pressure at Fill on 1/13/2017 in
Pellston from Arduino Data [Pa]
    T_pell_113 = 255.372            % Temperature at Fill on 1/13/2017 in
Pellston from Arduino Data [K]
    Pws_pell_113 = (2.718^(77.3450+0.0057*(T_pell_113)...
                        -(7235/T_pell_113)))/(T_pell_113^8.2);
    x_pell_113 = 0.62198*(Pws_pell_113)/(P_pell_113-Pws_pell_113);
    rho_pell_113 = ((P_pell_113/(R_dry_air*T_pell_113))*...
        (1+x_pell_113))/(1+(x_pell_113*(R_wet_air/R_dry_air)));     % Air
Density on 1/13/2017 in Pellston [kg/m^3]
    n_pell_113 = (P_pell_113*V_h)/(m_He_C*R_He*T_pell_113)        % Number of
Cylinders needed [cyclinders]
    rho_he_pell_113 = P_pell_113/((R_He*T_pell_113));               % Helium
Density in Helikite on 1/13/2017 in Pellston [kg/m^3]
    Fb_pell_113 = ((rho_pell_113)*grvty*V_h)*Ntolb % Force from boyancy force
from helium [lbf]
    Fb_pell_113_N = ((rho_pell_113)*grvty*V_h);
    ExcessL_pell_113 = Fb_pell_113 - W - m_He_C*n_pell_113*kgtolb
    ExcessL_N_pell_113 = Fb_pell_113_N - W_N - m_He_C*n_pell_113;
%% Interested Launches

% Standard Atmospheric Data
    Loc_mat = ['Bos' 'UMBS' 'Kzoo' 'Harv' 'Las' 'Albu' 'Den' 'CPER'];
    DUM_Loc_mat = transpose(Loc_mat);
    alt_mat = [9 233 290 332 1428 1619 1639 1645]  % Altitudes of Launches
[Bos,UMBS,Kzoo,Harv,Las,Albu,Den,CPER]
    DUM_alt = transpose(alt_mat);
    Dum_mat = tsa(alt_mat,'si');                              % Dummy Matrix of
TSA information
    P_mat   = Dum_mat(:,1);                                   % Column of
Pressures [Pa]
```

```
    T_mat   = Dum_mat(:,2);                                    % Column of
Temperatures [K]


        Pws_mat = (2.718.^(77.3450+0.0057*(T_mat)-7235*(T_mat.^-
1))).*(T_mat.^8.2).^-1;
        x_mat   = 0.62198.*(Pws_mat).*(P_mat-Pws_mat).^-1;

        rho_mat = ((P_mat/(R_dry_air*T_mat))*...
        (1+x_mat))/(1+(x_mat*(R_wet_air/R_dry_air)));

        n_mat   = (P_mat*V_h)/(m_He_C*R_He*T_mat);              % Number of
Cyclinders Needed to Fill TEA
        n_mat_red = n_mat(:,1)                                 % Column of
Number of Cyclinders Needed to Fill TEA

        rho_he_mat = P_mat/(R_He*T_mat);                        % Density of
Helium [kg/m^3]
        rho_he_redmat = rho_he_mat(:,1);                        % Column of
Density of Helium [kg/m^3]

        Fb_mat_N = ((rho_mat)*grvty*V_h);         % Boyancy Force Produced
[N]
        DUM_Fb_mat_lb = ((rho_mat)*grvty*V_h)*Ntolb;  % Boyancy Force
Produced [lbf]
        Fb_mat_lb = DUM_Fb_mat_lb(:,1)

        DUM_ExcessL_mat_N   = Fb_mat_N-W_N - m_He_C*n_mat;
        ExcessL_mat_N   = DUM_ExcessL_mat_N(:,1);
        DUM_ExcessL_mat     = DUM_Fb_mat_lb-W - m_He_C*n_mat*kgtolb;
        ExcessL_mat         = DUM_ExcessL_mat(:,1)
        format shortG
        X = [DUM_alt P_mat T_mat n_mat_red Fb_mat_lb ExcessL_mat]


% May Averages from Underground Weather.com
    % Input Data
        P_may = [101389 98815 98070 97562 85608 83136 83136 83272]  % Station
Pressure [Pa]
        DUM_P_may = transpose(P_may);
        T_may = [15 12 15 14 20 18 12 13]+273                       % Mean
Temp. [K]
        DUM_T_may = transpose(T_may);
    % Calculations
        Pws_may = (2.718.^(77.3450+0.0057*(T_may)-7235*(T_may.^-
1))).*(T_may.^8.2).^-1;
        x_may   = 0.62198.*(Pws_may).*(P_may-Pws_may).^-1;

        rho_may = (P_may.*(R_dry_air.*T_may).^-
1).*(1+x_may).*(1+x_may.*(R_wet_air/R_dry_air)).^-1;

        DUM_n_may   = (P_may*V_h).*(m_He_C*R_He*T_may).^-1             %
Number of Cyclinders Needed to Fill TEA
        DUM2_n_may       = transpose(DUM_n_may);
        n_may       = DUM2_n_may(:,1);
```

```
        rho_he_may = P_may.*(R_He*T_may).^-1;                                  %
Density of Helium [kg/m^3]

        DUM_Fb_may_N = ((rho_may)*grvty*V_h);          % Boyancy Force
Produced [N]
        DUM_Fb_may_lb = ((rho_may)*grvty*V_h)*Ntolb;  % Boyancy Force
Produced [lbf]
        Fb_may_N = transpose(DUM_Fb_may_N);
        Fb_may_lb = transpose(DUM_Fb_may_lb)

        DUM_ExcessL_may_N   = Fb_may_N - W_N - m_He_C*n_may;
        ExcessL_may_N       = DUM_ExcessL_may_N(:,1);
        DUM_ExcessL_may     = Fb_may_lb-W - m_He_C*n_may*kgtolb;
        ExcessL_may         = DUM_ExcessL_may(:,1)
        format shortG
        Y = [DUM_alt DUM_P_may DUM_T_may n_may Fb_may_lb ExcessL_may]

% May Average Maximum Temps w/ Same Station Pressures
    % Input Data
        P_max = [101389 98815 98070 97562 85608 83136 83136 83272]  % Station
Pressure [Pa]
        DUM_P_max = transpose(P_max);
        T_max = [19 20 21 20 28 25 19 20]+273                        % Mean
Temp. [K]
        DUM_T_max = transpose(T_max);

    % Calculations
        Pws_max = (2.718.^(77.3450+0.0057*(T_max)-7235*(T_max.^-
1))).*(T_max.^8.2).^-1;
        x_max   = 0.62198.*(Pws_max).*(P_max-Pws_max).^-1;

        rho_max = (P_max.*(R_dry_air.*T_max).^-
1).*(1+x_max).*(1+x_max.*(R_wet_air/R_dry_air)).^-1;

        DUM_n_max   = (P_max*V_h).*(m_He_C*R_He*T_max).^-1;             %
Number of Cyclinders Needed to Fill TEA
        n_max       = transpose(DUM_n_max)
        rho_he_max = P_max.*(R_He*T_max).^-1;                            %
Density of Helium [kg/m^3]

        DUM_Fb_max_N = ((rho_max)*grvty*V_h);          % Boyancy Force
Produced [N]
        DUM_Fb_max_lb = ((rho_max)*grvty*V_h)*Ntolb;  % Boyancy Force
Produced [lbf]
        Fb_max_N = transpose(DUM_Fb_max_N);
        Fb_max_lb = transpose(DUM_Fb_max_lb)

        DUM_ExcessL_max_N   = Fb_max_N-W_N - m_He_C*n_max;
        ExcessL_max_N       = DUM_ExcessL_max_N(:,1);
        DUM_ExcessL_max     = Fb_max_lb-(W+ m_He_C*n_max*kgtolb);
        ExcessL_max         = DUM_ExcessL_max(:,1)

        format shortG
        Z = [DUM_alt DUM_P_max DUM_T_max n_max Fb_max_lb ExcessL_max]
```

```
%%  Plots
     figure(1)
        [AX,H1,H2] = plotyy(alt,p,alt,t);
        grid on
            Kzoo = line([290 290],[0 1000000],'Color','r'); %Kalamazoo
Elevation [m]
            line([1645.0,1645.0],[0,10000000],'Color','k'); %CPER Elevation
[m]
            line([332.1,332.1],[0,10000000],'Color','m');   %Harvard Forest
Elevation[m]
            line([233.0,233.0],[0,10000000],'Color','k');   %UMBS Elevation
[m]
            line([1428.0,1428.0],[0,10000000],'Color','k'); %Las Cruces
Elevation [m]
            line([1618.5,1618.5],[0,10000000],'Color','m'); %Albuquerque
Elevation [m]
            line([1638.7,1638.7],[0,10000000],'Color','b'); %Denver Elevation
[m]
            line([8.9,8.9],[0,10000000],'Color','c');       %Boston Elevation
[m]
        xlabel('Altitude [m]')
        ylabel(AX(1),'Pressure [Pa*10^4]')
        ylabel(AX(2),'Temperature [K]')
        title('Pressure,Temperature vs. Altitude using TSA Code')
        legend('Pressure','Kalamazoo = 290','CPER = 1645','Harvard =
332','UMBS = 233','Las Cruces = 1428','Albuq. = 1618','Denver = 1638','Boston
= 8','Temp.')
```