LIGHTER-THAN-AIR ROBOT SWARM

A Design Project Report

Presented to the School of Electrical and Computer Engineering of Cornell University in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering, Electrical and Computer Engineering

Submitted by

Hongxi Jin (HJ439), Shiqi Li (SL2454), Tongqing Zhang (TZ422), Yingjie Zhao (YZ483), Vincent Lu (YL929)

MEng Field Advisor: Joe Skovira

Degree Date: May 2022

Table of Contents

Abstract

Master of Engineering Program
School of Electrical and Computer Engineering
Cornell University
Design Project Report

Project Title: Lighter-Than-Air Robot Swarm

Author: Hongxi Jin (HJ439), Shiqi Li (SL2454), Tongqing Zhang (TZ422), Yingjie Zhao (YZ483), Vincent Lu (YL929)

Abstract:

As higher buildings are becoming more and more common, the chances of an emergency happening are increasing. As a fire is going to occur 50 stories above the ground when vision is blurred, the surrounding is filled with smoke and darkness. Without an organized evacuation plan set in place, the aftermath can be devastating. Introducing the Lighter Than Air project, a swarm of blimp drones to guide people in need to safety in a safe manner. It is installed in hallways, and if an emergency happens, it deploys and guides nearby people to the nearest exit. It knows the floor plan to find the nearest exit, and it has LED onboard to shed light on the ground and point the direction. It has three motorized propellers giving its three degrees of freedom. The blimp swarm communicates with each other through WiFi, if they meet at one intersection, some of the blimps would exit the main group and can stay behind in case others are in need. We continued the previous team's work, improved their prototype, and implemented the payload logic board with a more compact, lighter design. We also replaced the Apriltag with Aruco Tag for a more generalized setup environment and faster recognition. Finally, we successfully completed the production of a single blimp and the initial flight test through manual control.

Keywords: evacuation, swarm, drone, blimp robot, embedded system.

Project Contribution

Hongxi Jin (hj439):

- Deployed and tested AprilTag in local desktop
- Fixed bugs for the build of AprilTag detection code
- Configured the Linux environment for AprilTag detection
- Developed the Aruco Marker control functionality to realize self-navigation
- Helped with Aruco Markers detection test on board

Shiqi Li (sl2425):

- Troubleshooted the base station program and fixed logic errors
- Set up the corresponding environment required by the program
- Installed necessary libraries and resolved the dependency problems for base station
- Modified the data structure for image detection data based on ArUco Tags and visualized data on the user interface of base station client

Tongqing Zhang (tz422):

- Investigated and fixed WiFi communication issue between base station and the drone
- Investigated and fixed the bug that freeze the base station client
- Tested and maintained modules on the motherboard of the drone including control system, data transmission (I2C) and hardware connections
- Tested and improved power system of the drone
- Tested functionalities of the drone including LED, motor drivers, and Time-of-Flight sensors
- Built the blimp mounted on a double-layer ballon “stack” that can balance in the air
- Implemented manual flight control by commands from base station

Yingjie Zhao (yz483):

- Investigated PCB circuit diagram and model number of required parts, made corrections on it
- Tested correctness and connectivity between the custom PCB and the Raspberry Pi Zero, ensured stable signal transmitting
- Constructed and soldered a new PCB with better parts according to the circuit diagram
- Designed and implemented more reasonable parts layout and wiring, including the location of DC motors, ToF sensors and power booster
- Innovatively connects the PCB to the Raspberry Pi Zero through straight male pins soldering, greatly reduced the number of jumper wires and decreased the overall weight
- Designed and printed motor shields using 3D modeling according to the size of PCB, motor and propeller
- Helped with blimp build and flight test via manual control

Yuanli Lu (yl929):

- Researched control algorithms and initialized a model of LQR algorithm
- Helped with Apriltag implementation
- Migrated Apriltag to Aruco Markers for image detection system

Executive Summary:

Our project is based on the work of the previous team. Though we had our vision, replicating previous work was a challenge for us. The challenge of replicating lies in connecting the design through the PCB circuitry, and recreating the execution environment of the payload as well as the base station. With limited documentation, we were able to reproduce the last team's result but with a semester's time. Thus, we still have been heavily referencing the previous team's report on which our work is based, which provides a refined version for future team's reference.

By the beginning of the second semester, we were able to optimize the last team's prototype which was a proof of concept and performed a flight test on mylar balloons. In order to perform the flight test, we had to cut redundant weight, such as weight from connection cables and batteries, as well as to cut redundant computing power by reconfiguring the sampling rate on the payload, considering that the computing power was limited to only one Raspberry Pi Zero.

In the end, we achieved a flight test by a brand-new blimp, which is discussed in this report. As we fixed some issues from the previous team's legacy hardware, we also achieved propulsion and manual command control, while some of the initial design limited our ability and needed to be revisited. Nevertheless, our approach was a huge step forward as we implemented and optimized the initial design proposed by the previous team, which could also provide considerable convenience and guidance for future teams as a pretty good reference.

The main achievements and architecture of the project are shown below:

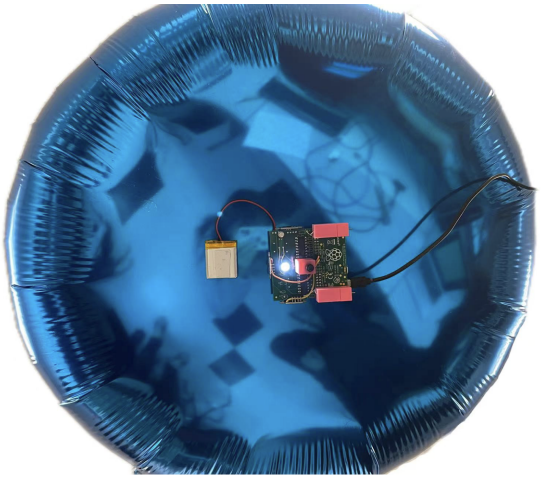

Fig.1 Below View of the Blimp

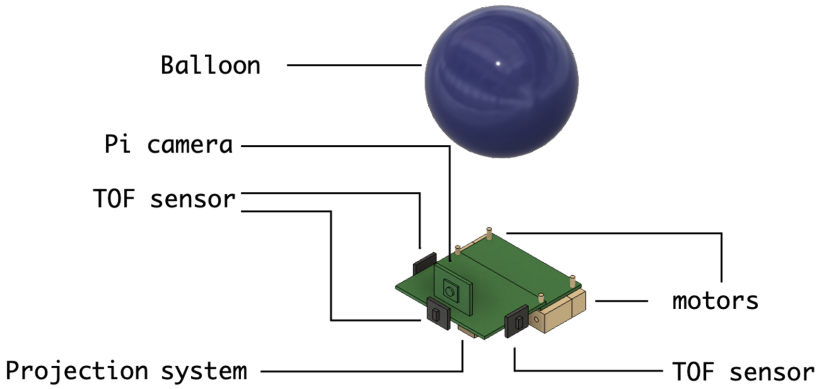

Fig.2 Architecture of the Blimp

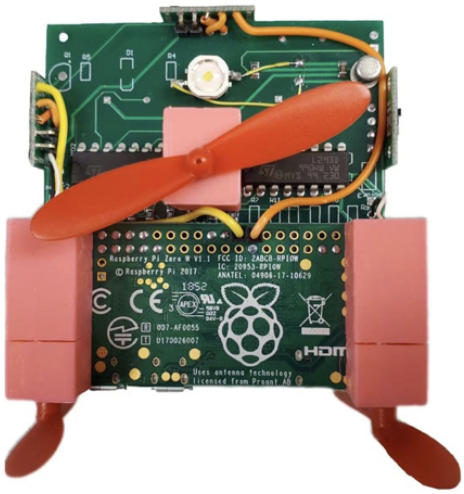

Fig.3 PCB View from Below

1. Introduction:

As higher buildings are becoming more and more common, the chances of an emergency happening are increasing. As a fire is going to occur 50 stories above the ground when vision is blurred, the surrounding is filled with smoke and darkness. Without an organized evacuation plan set in place, the aftermath can be devastating. The Lighter Than Air project introduced a swarm of blimps based on drones mounted on balloons capable of guiding and evacuating people in need in a safe manner. It is installed in hallways, and if an emergency happens, it deploys and guides nearby people to the nearest exit. It knows the floor plan to find the nearest exit, it has LED onboard to shed light on the ground and direct the way. It has three motorized propellers giving it three degrees of freedom. The blimp swarm communicates with each other through WiFi. If they meet at one intersection, some of the blimps would exit the main group and stay behind in case others are in need.

With the given motivation described above, we aim to build an evacuation device that is capable of self-navigation and swarm interaction. For the first semester, we reimplemented the last team's prototype design. It took us some time to figure out their designs and intentions for every feature and the pin logistics. The ramp-up curve is pretty steep because some of the technologies are new to us, such as the python socket package and the wifi protocol, etc. In the second semester, after we figured out the previous team's design, we started implementing our own design which can be divided into these 4 sections:

1. Streamline the prototype design and cut weight where it can.
2. Integrate the hallway with the navigation system, assigning Apriltags to each section.
3. Integrate our design into mylar balloons and perform a stable flight test via manual control.
4. Implement autonomous self-navigation based on a planned trajectory.

2. System Design:

Through the investigation and reimplementation of the previous design in the first semester, we have greatly improved our understanding of the overall design and goals of the project while also uncovered many issues not listed in the previous project report. So we conducted a more in-depth study of them and proposed the following feasible designs.

2.1 PCB Optimization

The PCB prototype from the previous year's team provides us a lot of help. Based on that, we found out notations printed on the PCB are wrong, but this does not affect the functionalities of the PCB. Besides that, there are two things wrong that affect functionalities, one is the distance of the soldering pad for the inductor should be 6.25 mm instead of 2.5mm; also for the LED on the back side of the PCB, the distance of the soldering pad for the LED should be 7.40 mm instead of 4.0 mm.

To solve both problems, we use a pitch adapter to fit the inductor, then use a double sided tape to stick it on the PCB, and use 30 AWG wire to connect it to the circuits.

Fig.4 Inductor Connection

Do the same operation for the LED connections, but without the pitch adapter, since we do not need it. Simply solder LED legs to the PCB using the 30 AWG wire, and stick it onto the PCB using double sided tape.

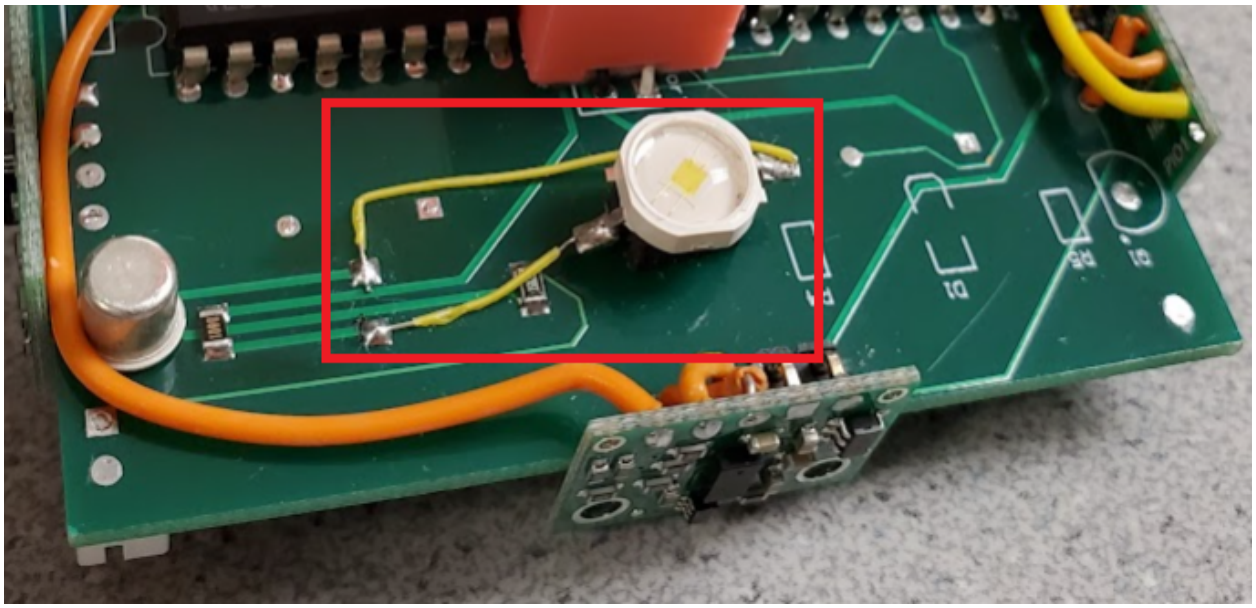

Fig.5 LED Connection

2.1.1 Circuit Debug

There is an error at the LED circuit part, but the connections are correct on the actual soldered board. The NPN transistor notation on the schematic should be flipped horizontally. The current flow is from the “Booster (5V)” to “GND”; therefore, the arrow on the NPN transistor should point to the GND.

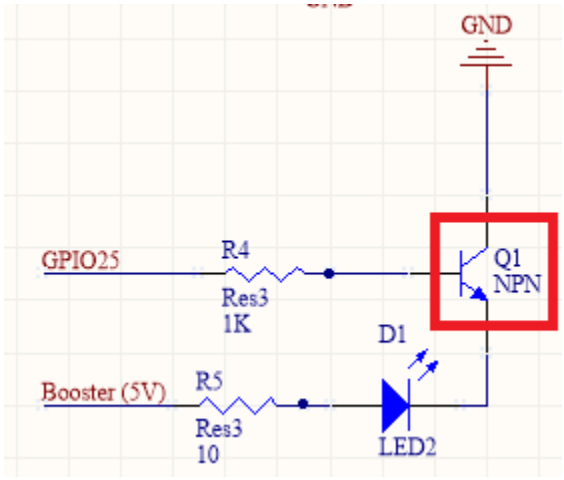

Fig.6 Incorrect Direction of the NPN Transistor

For both I2C resistors that connect in serial (SDA and SCL) to three ToF sensors, according to the circuit schematic, it states 1K ohms, but since every ToF sensor itself contains I2C resistors on board, we found out there is no need to add extra resistors on the PCB; otherwise, it will cause I2C signal decay issue, so we replaced them by 0 ohms SMD resistors.

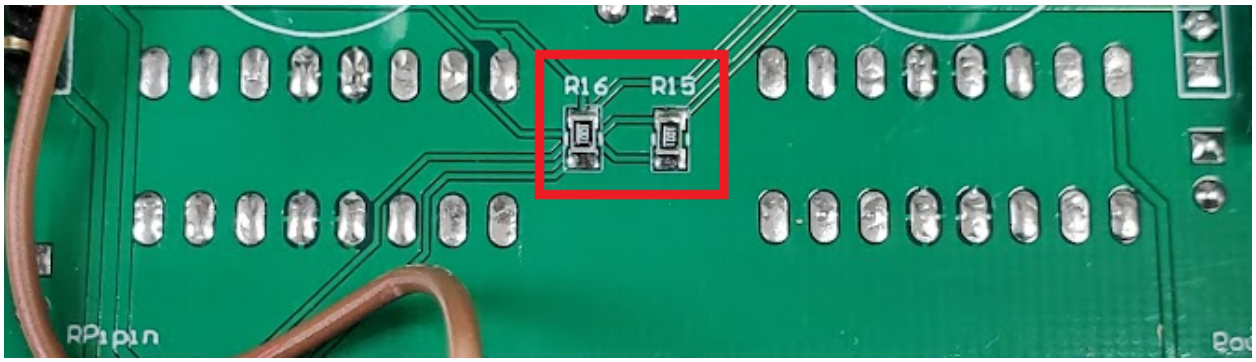

Fig.7 I2C Resistors on the PCB

2.1.2 Connections Between the PCB and Raspberry Pi Zero

Our goal is to decrease the total weight and increase overall rigidity. We thought we could get rid of those Dupont wires between the custom PCB and the Raspberry Pi Zero. Also, rework on soldering to make a clean and tidy look. Our plan is to use a female socket and male pins to replace Dupont wires, in order to reach our minimal design idea.

2.2 Power Requirements

According to the previous design, we tried to use only one Lithium Ion battery (3.7V 650 mAh) to power the PCB and the Raspberry Pi Zero in the meantime. However, during operation, the Raspberry Pi Zero often restarted due to insufficient current flow, leading to unstable. So we changed the design and tried to use a dedicated battery to power the PCB, while using another battery to provide 5V through a PowerBooster 500 board that connected to the Raspberry Pi Zero by using the GPIO pins #4 and #6.

2.3 Detection System

Tag-based detection is for the blimp to recognize its environment, and also to recognize other blimps in the swarm. So it is for the blimp operating logistics. It is very simple to use and doesn't need a lot of computing power for the blimp to operate on it.

As the previous team was using Apriltags for this task, we transferred the system to Aruco tags. While Apriltag uses fewer resources and is faster, it is less accurate. We now designed the tag detection to operate every few seconds, because image processing at every frame is still too computationally intensive for a Pi Zero, and the blimp is moving very slow, so we don't need it to recognize tags very often, though Apriltag takes fewer resources, the computational resource isn't a problem now. Because now we take a shot every few seconds, the accuracy needs to be guaranteed, which Aruco tags provide. Also, the Apriltag environment is hard to set up for every Pi Zero, and we plan to have a swarm of blimps, it is hard to setup and maintain each Pi Zero. Aruco is built-in with opencv. Considering these factors, Aruco markers are more suited for our design.

2.4 Position Adjustment of Motors and Propellers

The previous group first glued two motors to the sides of the PCB and then glued a motor to the underside of the PCB using hot melt adhesive to provide the drone with propulsion in different directions. However, the stability of this design is insufficient, and there is a risk of falling off after a certain period of placement, causing irreversible damage to the PCB. Also, since the motor is in direct contact with the hot melt adhesive, the heat from the motor may cause it to melt and fall off.

So we designed shields for DC motors in different positions through 3D modeling, and hooked them into the screw hole of the Raspberry Pi Zero PCB.

2.5 Blimp Envelope

Based on the design by the previous team, we tested the payload of a single helium balloon in the real world and found that it was insufficient to provide needed lift for the entire system which is consistent with previous conclusions (See Appendix C). So we followed the idea proposed by the previous team that tried a multi-layer balloon "stack"

design which enabled us to gain greater lift by stacking two helium balloons.
When it comes to the choice of balloons, we chose a Frisbee-shaped balloon with relatively flat upper and lower sides, which is more conducive to the stacking of two balloons and the mounting of the drone itself.

2.6 Base Station

The base station is one of the most important parts since the base station is in charge of the communication between the software and the hardware as well as the communication between the blimp and the user controlling the blimp robot. The base station is designed as two parts: the client that runs on a local computer and the server that runs directly on the RPi. The client maintains the connection to the server using a defined object of the class "Conn" in the code. When initializing an object of "Conn", the Python Socket library is utilized to create a socket object and to connect the client to the server over WiFi. The client sends requests to the server while the server is able to respond to the client. The messages sent are encoded as strings. There are basically three modes of control planned by the previous team: manual mode, waypoint mode, and autonomous mode, but only the fully-manual mode is implemented. In our project, since we rely on Aruco tag information to plan routes and navigate the blimps autonomously, we put the waypoint mode aside and mainly focus on the autonomous mode. Using PyGame library, a GUI is implemented for the convenience of users who can view localization information of the blimp as well as recognized ID numbers from Aruco tags sent back from the client. The appearance of the GUI based on the previous team's work does not change much, though the interaction between the client and the server has been modified. Following figures show the GUIs of the base station in three modes according to the previous team's design [1].

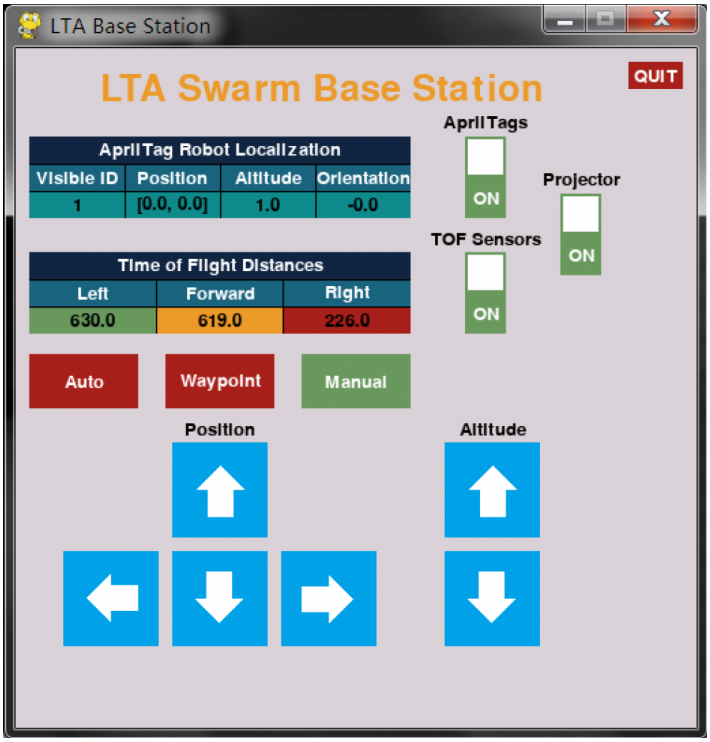

Fig.8 Manual & Autonomous Mode [1]

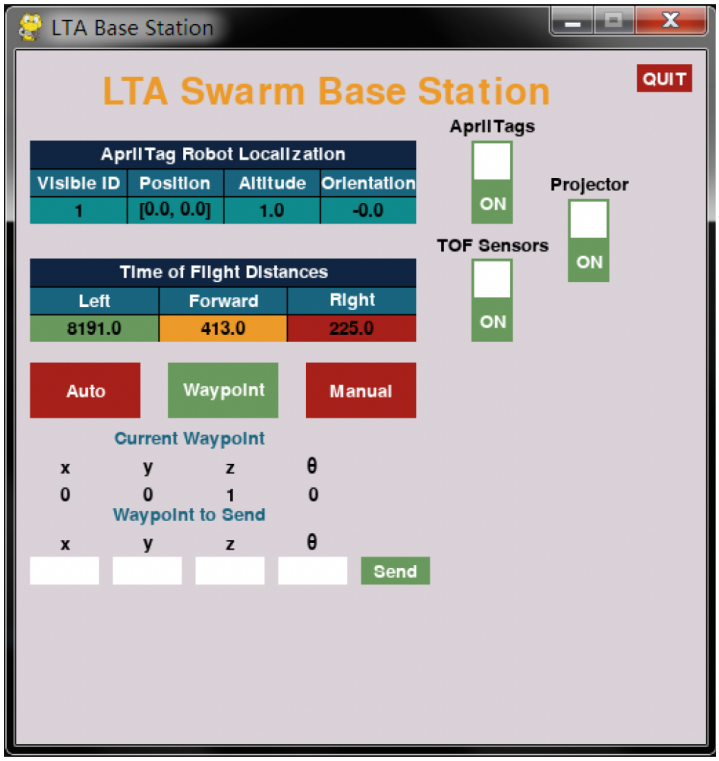

Fig.9 WayPoint Mode [1]

3. Implementation

3.1 Hardware Modification

Based on the investigation and tests in the first semester, we made the following changes to the hardware based on the optimized design: use the correct type capacitor, corrected the orientation of the NPN transistor, corrected the resistor value of the I2C channel, adjusted the position of the inductor and the LED, added a JST connector for easy battery connection.

3.1.1 Custom Power Booster Circuit on Board

The previous year's team mentioned the custom power booster circuit on the PCB doesn't work, and they believe that's because of the parts selecting problem. Our team did some research on the boost converter chip (MC34063A) and found out the previous team's assumption is correct. The input capacitor and the output capacitor must be low ESR (Erythrocyte Sedimentation Rate) types. Therefore we decided to use aluminum polymer capacitors made by Nichicon since they are low ESR type.

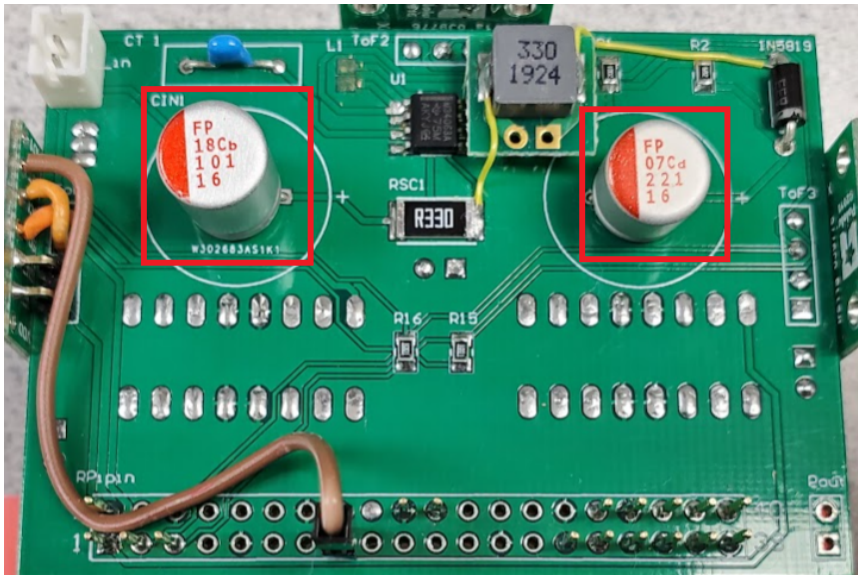

Fig.10 Low ESR Type Capacitors

3.1.2 Connection Between the PCB and Raspberry Pi Zero

Initially, we came up with an idea, which is to use a female socket and male pins to make a board-to-board perpendicular connection to each other, as the figure shows below. However, when we got all sockets and pins, we found out the socket itself has a height, which causes the total height increase, leading the center of gravity higher.

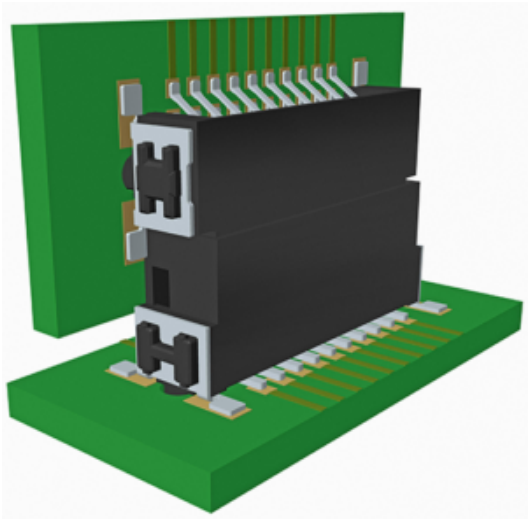

Fig.11 Board-to-Board Connection (Design)

Finally, we decided to use the simple way, use 2x20 straight male pins, put the custom PCB on one side and put the Raspberry Pi Zero on the other side and solder both sides together. The final outcome as the figure shown below, two boards connect at the same level, instead of perpendicular to each other.

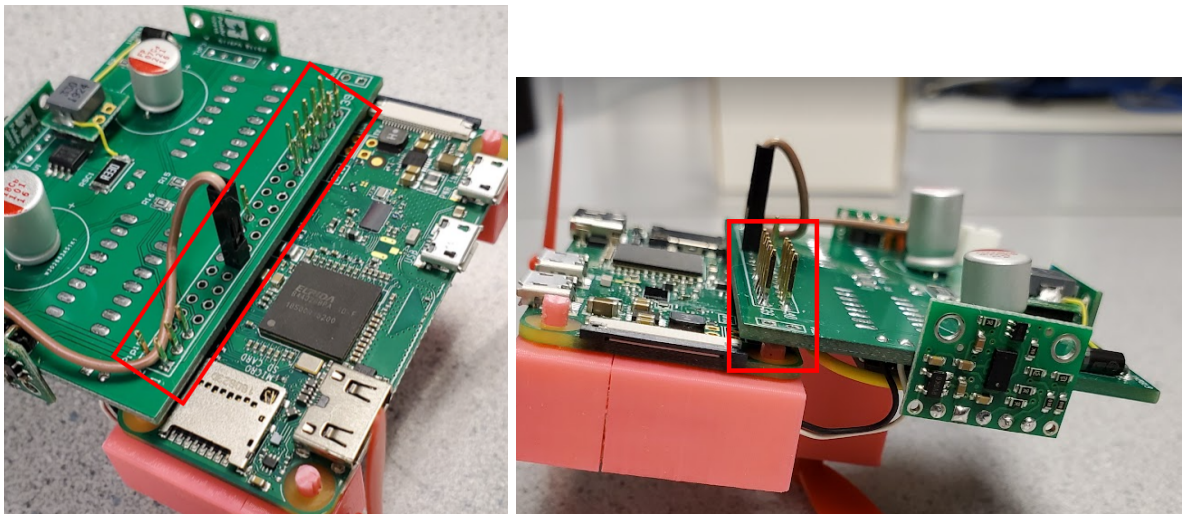

Fig.12 Board-to-Board Connection

3.1.3 Power System Improvement

Normally we power the Raspberry Pi Zero via the USB port, but the USB port is blocked by one of the rear propellers; therefore, we cannot plug a powerbank to it. We have to use batteries to power it, but one battery is not enough to power both Raspberry Pi Zero and sensors. So we use a PowerBoost 500 booster board, to raise the battery voltage from 3.7 V to 5V. And solder a female socket on the booster board, then plug it into the male pins on the Raspberry Pi Zero.

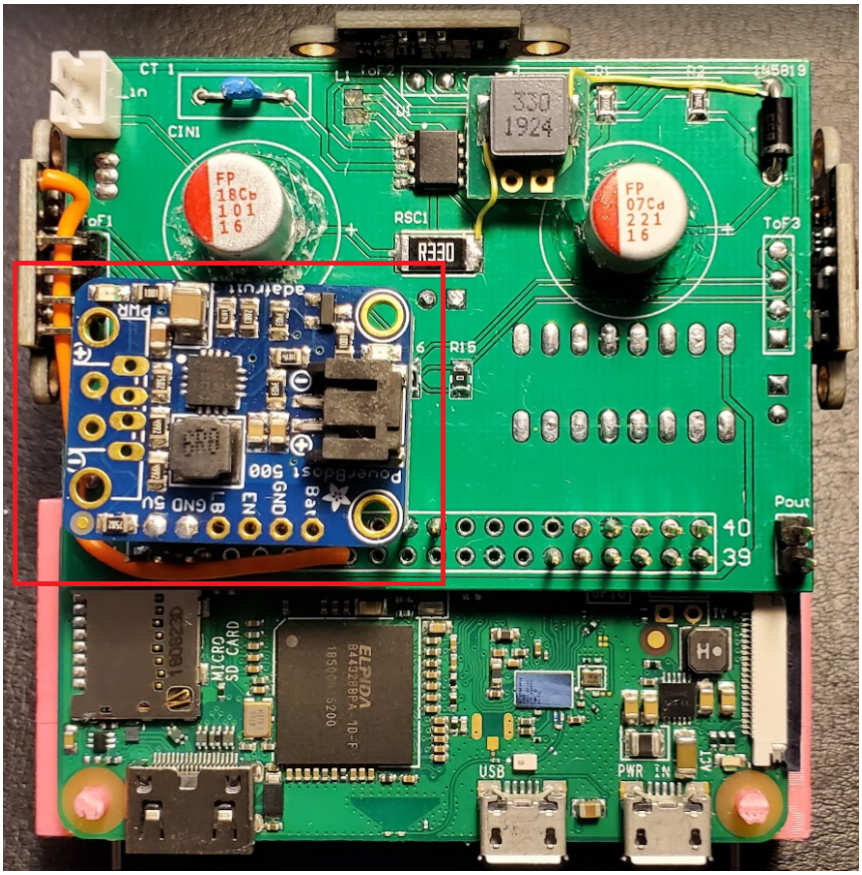

Fig.13 PowerBooster Board

3.1.4 Fabrication and Placement of Shells

Before building the protective shells for the motors, we accurately measured the length of the propellers and the inner diameter of the screw holes, and then completed 3D modeling and 3D printing according to the data to make motors stably stick on the PCB while ensuring that there is no collision between the propellers.

For the motor mounted under the PCB, we fixed it on the PCB using hot melt glue. Thanks to the protective shell, the contact area of the adhesion is larger and the motor

does not come into direct contact with the hot melt adhesive.

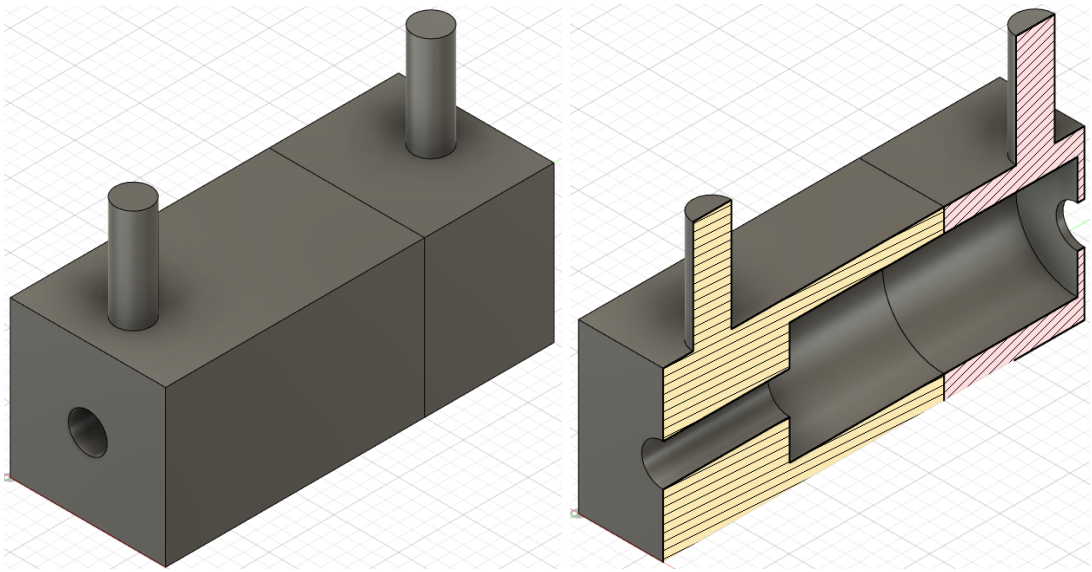

Fig.14 Shells for Rear Motors

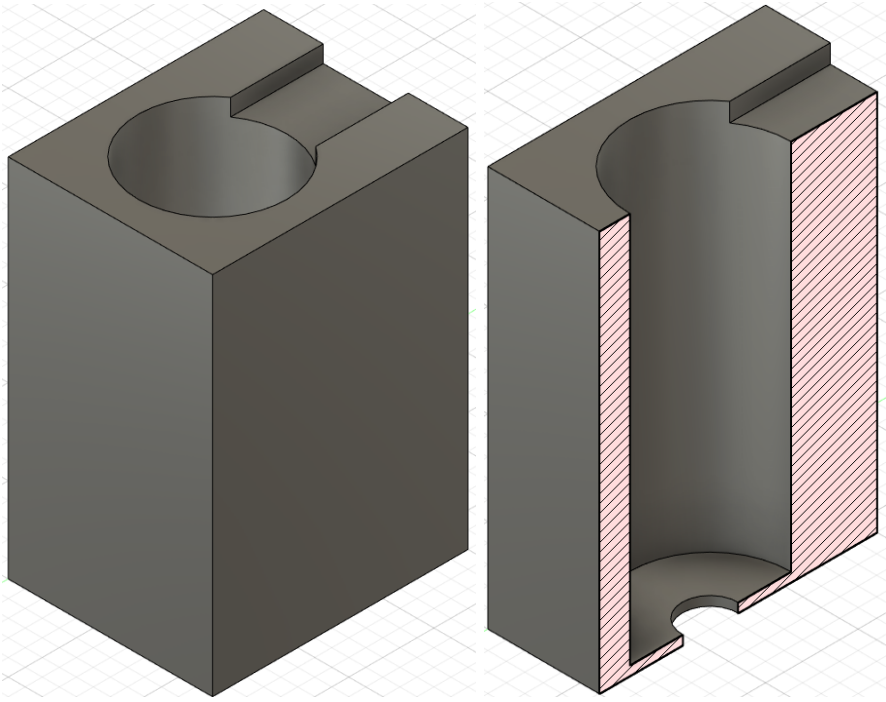

Fig.15 Shell for the Bottom Motor

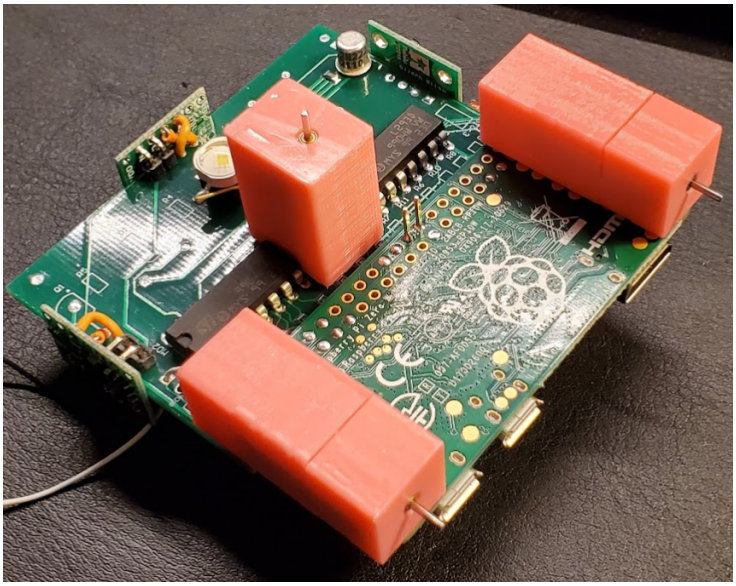

Fig.16 Install Shells onto the Board

3.2 Base Station

The Base Station program provides connections between all the modules, it communicates with, controls, and operates the flight computer which is a raspberry pi zero. It itself runs on a raspberry pi four. Since the code is embedded with the hardware, which needs a lot of feedback from the flight computer, there is a need to progress into re-implementing the hardware. It is easier to run the base station code on existing hardware than to simulate the environment.

To start to recover the functionalities of the base station which have been implemented by the previous team, the base station code is reviewed, intending to capture a whole picture of the base station's functionality.

Since the connection between the client and the server is based on TCP protocol, specific server IP addresses must be defined. In order to establish the connection, the IP address and the port number in the code is tried to be modified according to the IP address searched on the Raspberry Pi operating system. It is found out that both the IP addresses on the client and server side should be the server IP address. Another bug encountered when establishing the connection between the client and server is that when trying to open the file whose name is mistakenly spelled by the previous group. The file which should have been named "servercontrol.py" is mistakenly named "severcontrol.py", resulting in an error when reading the file. To prevent any further unexpected

errors when running the programs, the decision is made to maintain the file name "severcontrol.py" instead of changing the file name.

After the logic problems are troubleshooted and the code is debugged, multiple compilation errors still exist and remain to be detected in the base_station.py file so that the program can be compiled. During the compilation process, copious numbers of errors emerge which mostly are related to the environment setting problems, known as dependency issues. A lot of files were reported to be missing due to lack of corresponding dependencies. All the required files and packages suggested on the command line are manually downloaded so that necessary packages or libraries could be imported successfully when compiling the python file. Lacking necessary packages was a common problem when the functionalities of the blimp project were tried to be recovered. In another file named as "flightsensor.py", there is a library named "VL53L0X" imported while no such package existed in the project code given by the previous team. Thus, to fix the problem, corresponding packages are downloaded from Github. Particularly, a specific Makefile is written that is necessary to compile and launch the program since it is not given on Github.

Another compilation problem faced in the project is a type of comparison error. One thing worth noting in Python syntax is that using "is" in Python essentially means comparing addresses of two objects while using "==" in Python means comparing values of two objects. In the given version of code from the previous team, taking line 793 in "base_station.py" as an example, "**elif**(event.type **is** MOUSEBUTTONDOWN):" uses "is" to compare two event types which reported an error while compiling the program. There are six places requiring modifications and this problem is fixed by changing "is" to "==".

When the blimp detects and recognizes an ArucoTag, it is able to store the detection information into an array on the server side. The server communicates with the client over WiFi connection and transmits the information stored in the array to the client side. The goal is to decode the data in the array and print out the detection information on the base station GUI. The problem is that the decoding process fails. By reviewing the code, one can notice that the previous team created a high dimensional data array in the client side. The 2D array contains extra information including the coordinates of the blimp that are redundant in this project, causing trouble in translation of detection data. Since we are using ArucoTags to autonomously navigate the blimps, we could discard the coordinates. The solution to solve this problem is to modify the 2D

array into a single dimensional array, and detected ArucoTag information can be correctly translated and printed as visible ID on the interface.

After fixing the above errors and improving the AprilTag information decoding, the program is successfully compiled and a GUI of the base station can show up on the screen of the client side, which is on a local computer.

3.3 Tag-Based Detection

To realize the self-navigation of the blimp, one of the most important parts is localization.

The previous team proposed a self-navigation mode for the blimp, which applied AprilTag to realize localization, but they did not deploy it successfully.

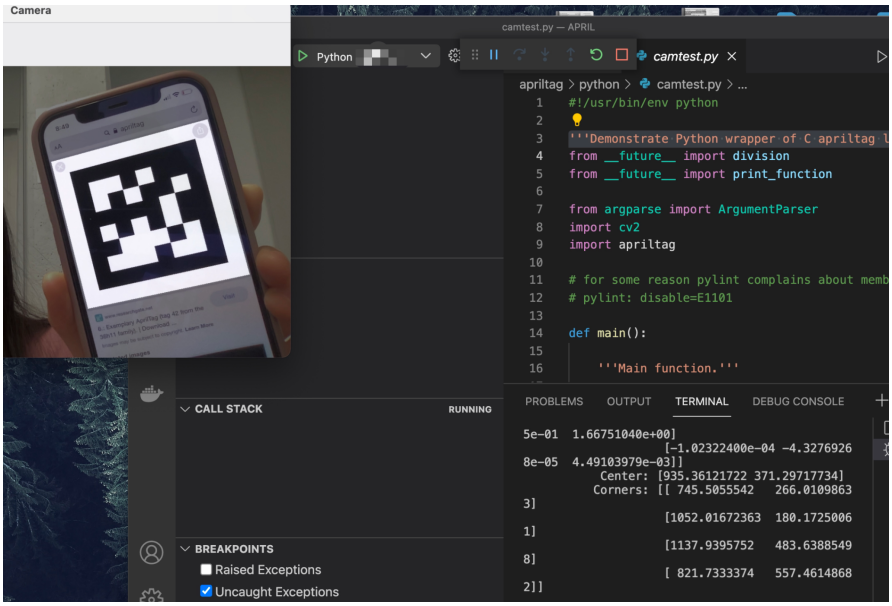

Fig.17 Apriltag Test

We tried implementing the Apriltag recognition library building from C. But due to the low version of Raspberry Pi, we are having trouble installing some of the packages locally. To solve this issue, we used a package called ArUco markers, which reads the same AprilTags but in a different way. Compared to AprilTag, ArUco markers are easier to set up, having higher accuracy, but it takes much more computational cost since the package is based on OpenCV. Also, the newer versions are no longer open-sourced, it has been licensed. Our task requires less computational work, if it can be run on flight, then we wouldn't need to transmit video data to the base station for processing. But on the other hand the environment may not be ideal, so accuracy

also needs to be guaranteed.

In the beginning, we were afraid that the higher computational complexity of ArUco would cause the higher latency of the system. And the previous team recommended to use tag family 16h5 instead of 36h11 (smaller tag family) and reduce the camera resolution to reduce the latency. In our implementation, We kept the 36h11 tag family and reduced the camera resolution. Also, we installed a light version of OpenCV, since it is quite a challenge to install the entire package with all its additional functions, and we only need the basic functionality of the package. What's more, we considered the following situations. When the camera detects more than one tag, which tag should be the one it wants to go for? In our algorithm, we output the four corner information of each tag and calculate the area of the tag based on these corner information. We estimate the one with the largest area as the "target tag", which means the tag is nearest to the blimp. Another situation is to detect the direction of tags, in other words, detect whether the target tag is to the left or right of the blimp. Based on the corner information, we calculate the ratio of left edge length to the right edge length. If the ratio is greater than the threshold we set, we determine the tag is in the left front of the blimp, and vice versa. Then we adjust the position of the blimp by giving the corresponding command.

After installation, we pre-defined the turning direction of each tag. For example, if the recognition result is tag ID 1, then the blimp would make a 90 degree left turn. If the tag ID is 7, then the blimp would make a 90 degree right turn. We integrated the detection code into the base station part.

3.4 Build the Blimp

Before connecting everything together, we first tested the balloons' payload to ensure the feasibility of the experiment. We tied a plastic bag to the balloons with a string and put the various components (including the drone body, a separable camera, two batteries) into it one by one to see if the balloon could balance in the air.

In experiments that lasted for several days, we found that as the balloon gas leaked over time, its payload would gradually decrease, which made it necessary to add some load (some small magnets) to the drone from the beginning to the need to re-inflate the balloon to ensure its buoyancy.

Then we used 3M Square Double Sided Foam (40mm x 40mm x 3mm) to connect the

two balloons and the drone body in sequence which is shown in Figure X, and used transparent glue to fix the two batteries as the power supply for the drone in front of the drone body on the bottom balloon.

In addition, since the connections between the parts are not rigid, and the density along with center of gravity of the balloon and the drone are different, it is crucial to adjust the position of the drone relative to the balloon to keep it level.

We first aligned the two balloons, connected their centers, and then also centered the drone body on the bottom balloon. At this time, due to the fact that the rear PCB of the drone has more weight, its center of gravity is not in its model center. So we used the two batteries connected to the drone to help adjust the overall center of gravity and direction level by fine-tuning its placement.

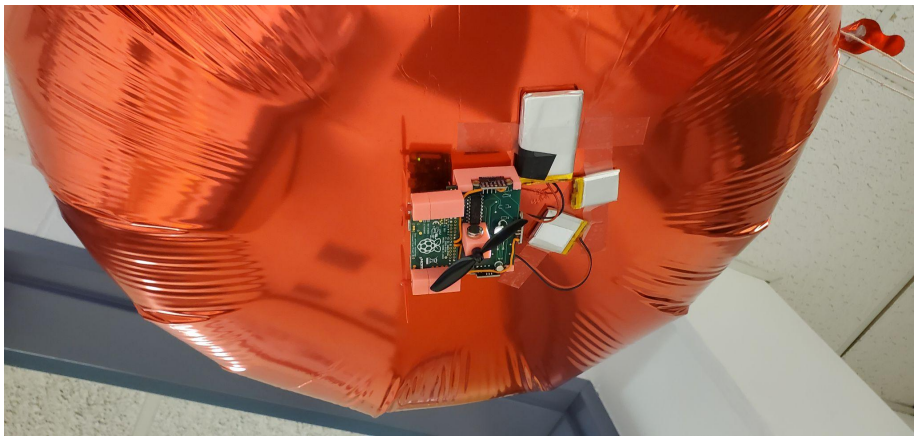

Fig.18 Bottom View of the Blimp

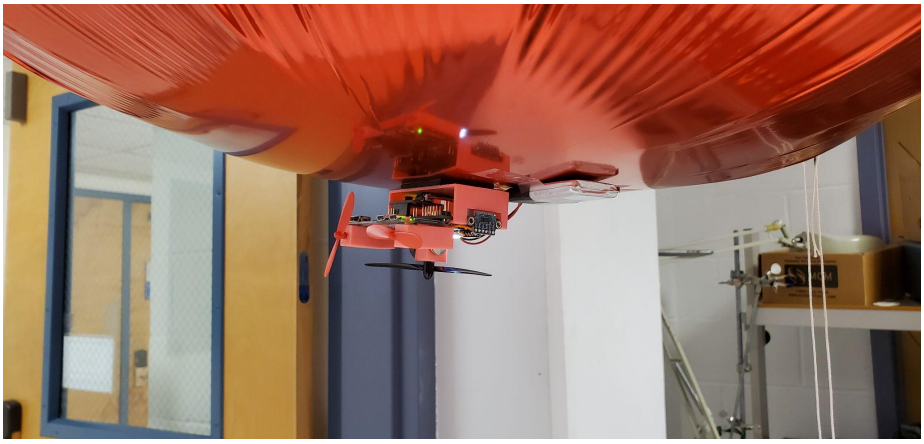

Fig.19 Side View of the Drone

Fig.20 Overall View of the Blimp

4. Test and Results

4.1 ArUco Test

Before connecting the hardware part to the balloon, we tested the functionality of the detection system. It recognized the tags in A4 size paper from about 30 centimeters to 2 meters, and the latency had little effect on the whole system. Then we tried to enlarge the detection range. To adjust the camera focus, we displayed the real time video stream and tested the focus distance for many times. Finally, we got the recognition distance of the tags from 20 centimeters to 2 meters. The range is not wide, but not too bad.

After connecting the hardware part to the balloon, we found that the moving speed was much lower than expected. Thus, we set a larger detection period to reduce the average computational complexity. We tested the detection period from 1 to 3 seconds, and set the detection period as 2 seconds. This period setting achieved the lower error rate with the less amount of computational time.

The demonstration image of AruCo Markers is shown below.

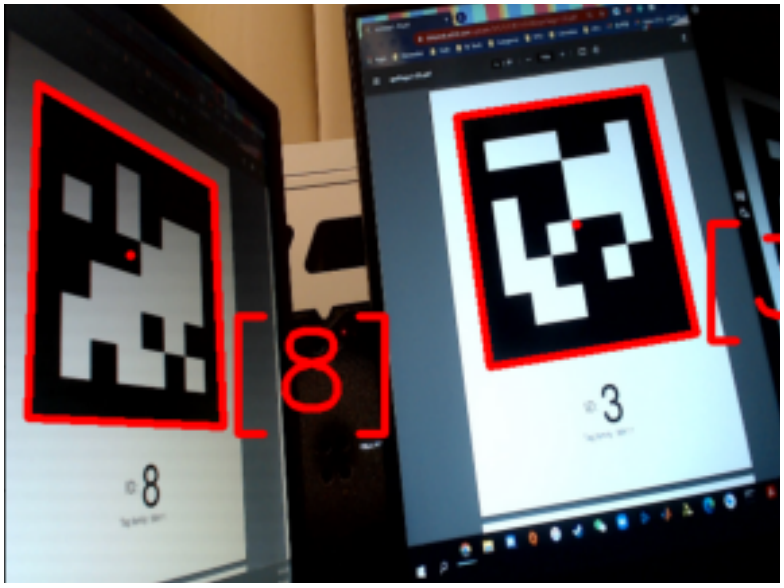

Fig.21 Aruco Marker Tests

4.2 Time-of-Flight Sensor Test

The base station is in charge of localization and detection information display. Therefore, the interface can be used to test the sensor and AprilTag detection program. When launching the base station program, each time a tag is recognized by the Pi camera, the recognized tag information is transmitted from the client to the server and is printed as the visible ID on the interface. By comparing the output information with the number represented by the tag, one can tell whether the detection function works properly. In addition, three sensors are tested by observing the changes of ToF data as shown in the following figures. When obstacles are placed near each sensor, ToF value decreases into a relatively small value, indicating the distance between the sensor and the obstacle.

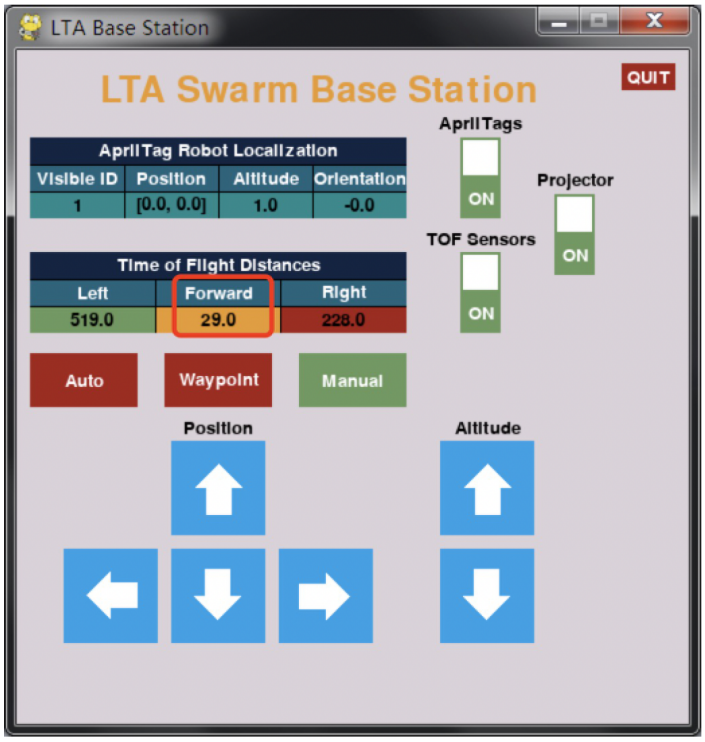

Fig.22 Testing on the Forward Sensor

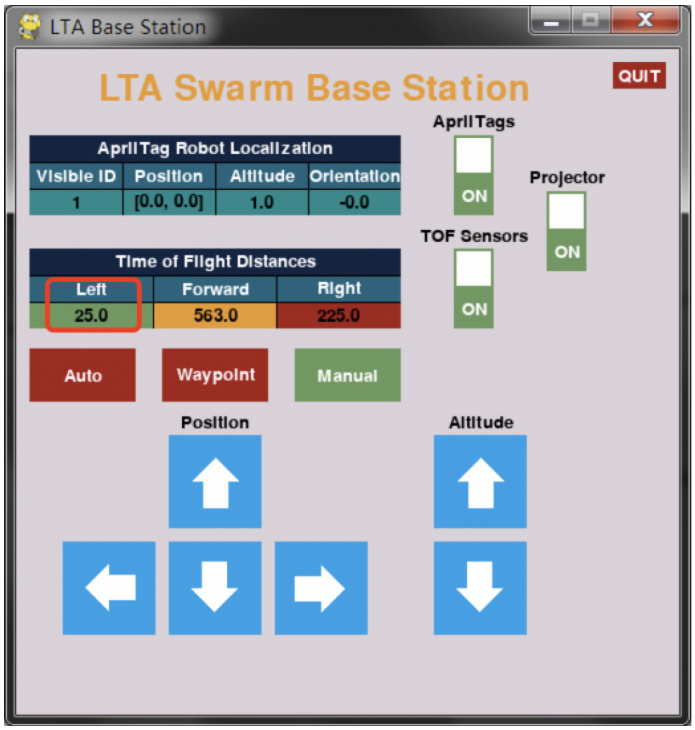

Fig.23 Testing on the Left Sensor

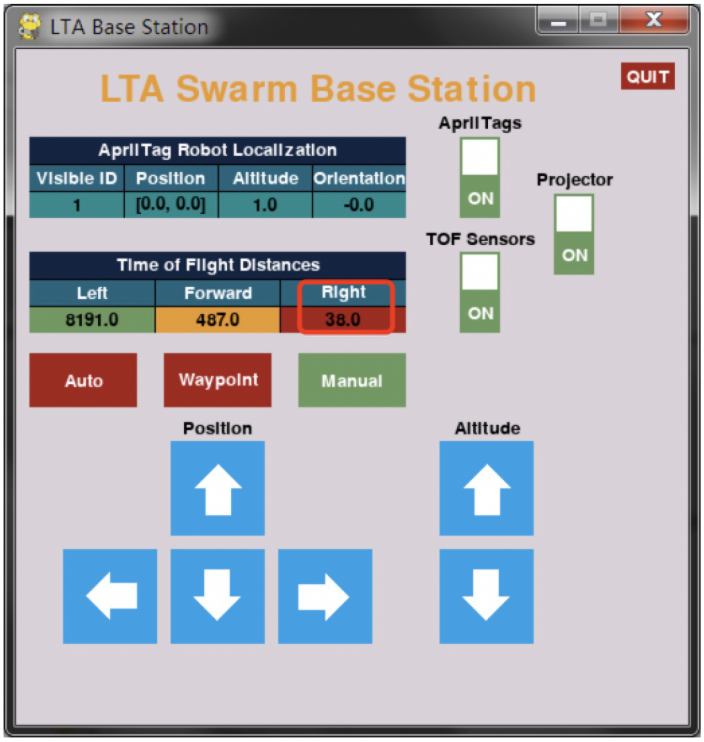

Fig.24 Testing on the Right Sensor

4.3 Motor Drivers Test

For the brand-new PCB we built and the motors housed inside shells that mounted on it, we tested their availability and stability both in hardware and software before flight test. For hardware testing, first we tested the correctness of the connections between pins on the PCB and motors using a multimeter, then individually tested the reliability of the high and low level control of these pins through GPIOs of the Raspberry Pi Zero to make sure that motors still work.

For software testing, we tested the overall operation logic of the motors, which are the main operation functions written in the drone system including forward, backward, turn left and right, ascent and dive. We ensured the consistency between commands and hardware responses during operation so that we were able to manually control the blimp remotely, which is also the basis for automatic navigation in the future.

During the test, we discovered and fixed some pin errors in functions, and optimized the control logic, so that the drone can obtain more torque when turning, performing faster and more sensitive control.

4.4 Flight Test via Manual Control

Having completed the basic tests for each function, we built a blimp by mounting the drone on a double-layer balloon “stack” for flight test via manual control in the real world.

In the experiment we tested and recorded the results of various operation commands listed below: forward, backward, turn left, turn right, float up, and dive.

Among these commands, as we adjust the weight to keep the blimp balanced in the air and use a propeller in a larger radius for the motor under the drone, the float up and dive commands worked quite well in a windless indoor environment, which allows the blimp to quickly adjust its altitude.

For the turning commands, since we optimized the control logic to make the motors distributed on both sides of the PCB rotate in opposite directions at the same time instead of using only one motor for propulsion, the blimp can get more torque when turning, performing faster and more responsive steering.

However, compared to the above commands, the forward and backward commands do not perform so well. Due to the lack of a direction maintaining system, the blimp cannot fly in a straight line due to the offset of the center of gravity, the uneven position, and the difference in the power of the motors on both sides, etc. After a short period, it would always deviate in a certain direction due to external influences or its own reasons, and then hit the wall, which would also be the main direction of improvement in the future.

Link to demo videos are below:

https://drive.google.com/drive/folders/13ZO07AMAKq9Th6wYsvI2vFa87eTBZEWe?usp=sharing

5. Known Issues

5.1 ArUco Detection

One problem is that we only have one pi-camera on board, without support for auto-focus. The tags aren’t always picked up because it isn’t focused enough. Adjusting focus to an adequate position, the recognition distance of the tags is from 20 centimeters to 2 meters. The range is not wide, which is restricted to scenarios. We tried to fix this issue by adding three different sizes of tag at each point so the Blimp can recognize it at

every point. But in the future, it is recommended to use a camera with auto-focus. Another issue is the paste position of tags. Ideally, the tags should be stuck opposite to the camera. However, the instability of the balloon would cause the vibration of the whole system. In some situations, the camera might detect more than one tag of nearly the same area. It is misleading. To solve this problem, we think about enlarging the distance of each tag to make sure that only one target tag is detected at a time.

5.2 Sensor Accuracy and Range

Since the range of the Time-of-Flight sensor is only about 1.2m, its reading will be fixed at the maximum range when the range is exceeded, so it can work relatively well in relatively narrow corridors, but it cannot accurately obtain the position of the blimp in wider corridors.

In addition, since the camera we used has a fixed focal length which required a certain distance from the pictures on the wall for tag-based image detection, sensor range that is too small may prevent the blimp from decelerating and stopping at the specified distance for image recognition, resulting in errors in the navigation system.

5.3 Flight Direction and Stability

In the flight test, we found that the blimp could not fly well in a straight line as well as turning a certain degree due to the lack of a direction detection system. Based on the inspiration from the fast robot class, we believe that an inertial measurement unit (IMU) sensor can be integrated in the drone to help solve the problems.

By integrating the angular velocity along the z-axis obtained by the gyroscope in the IMU sensor, the real-time angle of the blimp can be obtained to help keep a straight flight. By measuring the direction of the geomagnetic field by the magnetometer in the IMU, an absolute reference direction can be obtained to correct the flight angle of the blimp. In addition, the data obtained by the accelerometer in the IMU can be used to detect whether the blimp deviates from the course, and then anti-jamming can be carried out through LQR control.

6. Conclusion

We thoroughly investigated the design of the previous team, then made targeted improvements and optimizations, and built a brand-new lighter, more compact and highly integrated drone body. Having completed a series of functional and physical tests, we 3D modeled and printed tailor-made plastic components to protect the motor and hold the body, improving the reliability and stability of the drone.

Finally, we built a blimp that can balance itself in the air by attaching the drone to a double-layer balloon “stack” and successfully completed the flight test through manual command control in the real world. We believe this is a huge step forward as we managed to bring what we tested in the lab to the real world, which will also provide considerable convenience and guidance for future improvements.

7. Future Work

Since we have successfully built a blimp that can perform flight tests in the real world through manual command control, the next goal is to improve its flight stability, flexibility and control precision. We proposed to integrate an inertial measurement unit (IMU) sensor to optimize and solve these problems.

The gyroscope in the IMU can be used to obtain the real-time orientation of the blimp and keep the straight flight and precise turn given any degree via PID control. At the same time, the magnetometer in the IMU can be used to detect the direction of the geomagnetic field, which can be used to regularly correct the orientation offset calculated by gyroscope readings. The accelerometer in the IMU can detect the deviation of the blimp from the predetermined position and route, which can be then used for fine position correction through LQR control. In addition, replacing the Time-of-Flight sensor with higher accuracy and larger range ones will also greatly broaden available situations and improve anti-interference ability of the blimp.

As some future functionality has been implemented on the base station such as waypoint control, ArUco detection and automatic navigation but is not yet completed on the robot itself, providing more integrated functions for the drone such as PID control, LQR control, and more intelligent image detection is the main direction of future work. Ultimately, by integrating all the functions mentioned above, we hope to realize

autonomous self-navigation of the blimp based on a planned trajectory in a given map. Fig.25 shows the route we designed for the hallway of Philip Hall based on the map. The starting point is Phillips Hall room 239 and the destination is the north entrance of the Phillips Hall.

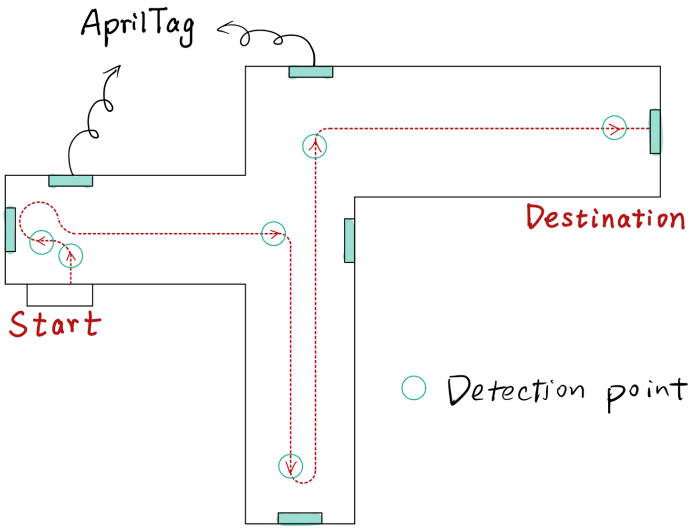

Fig.25 Planned Flight Trajectory

Acknowledgment

We would like to express our sincere thanks to our advisor, Professor Joseph Skovira for his continuous support and encouragement, who also provided his expertise in Linux Kernel and Craftsmanship helped us avoid lots of unnecessary trouble. Professor Skovira actively responded to our questions, made many constructive comments, and also provided convenient and quick help in the purchase of spare parts. We couldn't have made this far without his help and we sincerely appreciate all the help and encouragement we have received from him.

Reference

[1] Newport A, Wei J, Menon P, Dilip P. (2020). LIGHTER-THAN-AIR ROBOT SWARM. https://github.coecis.cornell.edu/tz422/Lighter-Than-Air/blob/main/reference/project_247_report.pdf

[2] Raspberry Pi Foundation. (n.d.). Raspberry Pi Zero W. Retrieved from https://www.raspberrypi.org/products/raspberry-pi-zero-w/

[3] Raspberry Pi Foundation. (n.d.-b). Camera Module V2. Retrieved from https://www.raspberrypi.org/products/camera-module-v2/+

[4] Adafruit Industries. (n.d.). PowerBoost 500 Basic - 5V USB Boost @ 500mA from 1.8V+. Retrieved from https://www.adafruit.com/product/1903

[5] ON Semiconductor. (2019). MC34063A, MC33063A, SC34063A, SC33063A, NCV33063A Datasheet . Retrieved from https://www.onsemi.com/pub/Collateral/MC34063A-D.PDF

[6] STMicroelectronics. (2018). VL53L0X Datasheet . Retrieved from https://www.st.com/resource/en/datasheet/vl53l0x.pdf

[7] Adafruit Industries. (n.d.-a). Adafruit DRV8833 DC/Stepper Motor Driver Breakout Board. Retrieved from https://www.adafruit.com/product/3297

[8] Texas Instruments. (2016). L293x Quadruple Half-H Drivers Datasheet . Retrieved from http://www.ti.com/lit/ds/symlink/l293.pdf

[9] Detection of ARUCO markers. OpenCV. (n.d.). Retrieved May 8, 2022, from https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html

[10] OpenCV. (2020). OpenCV. Retrieved from https://opencv.org/

[11] Multicomp. (2012). 1W High Power LED Datasheet . Retrieved from http://www.farnell.com/datasheets/1636581.pdf

[12] Broadcom LED Lighting series White, Cool 7250K 3.2V 350mA 140° 2-SMD, Gull Wing Exposed Pad . Retrieved from https://www.digikey.com/en/products/detail/broadcom-limited/ASMT-AW00-NUWK1/13906997

[13] Gibbs, K. (2020). Projector [Diagram]. Retrieved from

http://www.schoolphysics.co.uk/age11-14/Light/text/Projector/index.html

[14] PyGame. (2020). PyGame. Retrieved from https://www.pygame.org/

[15] Python Software Foundation. (2020). Socket. Retrieved from https://docs.python.org/3/library/socket.html

[16] Gupta, S. (2018, July 17). Calculating the Components' Values for Boost Converter [Table]. Retrieved from https://circuitdigest.com/electronic-circuits/3.7v-to-5v-boost-converter-circuit-diagram

[17] Python Software Foundation. (2020b). Threading. Retrieved from https://docs.python.org/3/library/threading.html

[18] Python Software Foundation. (2020c). Time. Retrieved from https://docs.python.org/3/library/time.html

Appendix A - Code Structure

Our code is located at https://github.coecis.cornell.edu/tz422/Lighter-Than-Air

Files needed on base station PC:

- base_station.py: code to run the base station GUI
- up_arrow.png, down_arrow.png, left_arrow.png, right_arrow.png: image files used in the base station GUI

Files needed on Raspberry Pi:

- Apriltag.py: code for running the Apriltags machine learning detection from the OpenCV optimized library
- Camera.py: function for initialize the camera setting
- Control.py: function for operating motors in different directions
- flightsensor.py: functions for operating and reading data from time-of-flight sensors
- LED.py: functions for operating the projector
- servercontrol.py: top level file for operating robot
- VL53L0X.py: library implementing the VL53L0X (TOF) platform-specific i2c functions through callbacks to the python SMBus interface
- Apriltag folder: contains library code needed for interpreting Apriltags

Appendix B - Base Station & Robot Control Code Installation

Installing and running robot code

1. Download required files for the robot listed in Appendix A to the Raspberry Pi
2. Install the time-of-flight sensor code by running the following command:
   ```
   pip3 install git+https://github.com/naisy/VL53L0X_rasp_python.git
   ```
3. Install OpenCV by running the following command:
   ```
   sudo apt install python3-opencv
   ```
4. Install and set up the AprilTags code by running the following commands in order:
   ```
   pip3 install dt-apriltags
   git submodule init
   git submodule update
   make build PYTHON_VERSION=3
   ```
5. To start the code, navigate to the directory containing servercontrol.py and run the following command:
   ```
   python3 servercontrol.py
   ```

Installing and running base station code

1. Download required files for the base station listed in Appendix A to a PC
2. Install NumPy by running the following command:
   ```
   pip3 install NumPy
   ```
3. Install PyGame by running the following command:
   ```
   python3 -m pip install -U pygame --user
   ```
4. Ensure that the variables `server_ip` and `server_port` on lines 757 and 759 of base_station.py match the IP address of the Raspberry Pi and the port number being used respectively
5. Make sure the server is already running on the Raspberry Pi
6. Run the base station code by running the following command from the directory containing base_station.py:
   ```
   python3 base_station.py
   ```

Appendix C - Weight Budget

Components	Mass Per Unit [g]	Quantity	Total Mass [g]
1. Raspberry Pi Zero W [1]	9.8	1	9.8
2. Power booster	3.1	1	3.1
3. Pi camera [2]	6	1	6
4. PCB + motors + flight-sensors	39.01	1	39.01
5. Lithium Ion Polymer Battery - 3.7v 2500mAh	43	1	43
6. Lithium Ion Polymer Battery - 3.7v 500mAh	10.5	1	10.5
Total			**111.41**