Sunset Industries
Drone Maze Traversal and Light Detection

Prepared by:

Sunset Industries

9366 Atlantic Street
Elizabeth, NJ 07202
734-123-1234

Chris Van Dyke, Samuel Hofmann, Hassan Kadiri,
Engineers in Training

Prepared for:

Robert Dick, President
Ariadne de Bothezat, Client

December 8, 2020

Table of Contents

Tables

Figures

Executive Summary

In our day-to-day lives, the sun's cycle tells us when to go to sleep, when to wake up, and when to do daily tasks. With this notion in mind, we could use this change between day and night to develop an algorithm for autonomous systems to make them more effective and efficient.

Ariadne de Bothezat, a client, contacted the president of our company, Robert Dick, with an interest in developing and marketing quadcopter platforms and sensor-electronic kits to enthusiasts and universities. As a result, President Robert Dick would like for all development teams to introduce to the client their autonomous quadcopter and its performance on an obstacle course. Our team, Sunset Industries, was required to develop a quadcopter that successfully and rapidly transverses an obstacle course and land in the landing zone while minimizing collisions. The algorithm that did the traversing should be correct and reliable in the presence of drifting conditions, and the PID controller parameters should enable a reliable, fast, and minimal collision flight process. Also, we were asked to develop and demonstrate our custom addition to the quadcopter.

The quadcopter is built with a combination of various electrical components. Our quadcopter uses 4 lidar sensors mounted on top of the quadcopter to determine its position relative to an obstacle around it. For this project, we decided to use a sequence algorithm to transverse the obstacle course. Based on its process, the algorithm will switch between moving forward, backward, left, or right. Our current quadcopter fulfills most of our client's requirements, however, it is not reliable in the presence of drifting conditions because it is a sequence algorithm. In the next generation of our algorithm, we would like to implement a dynamic algorithm that will be successful in traversing the maze, wherein changing conditions will not be an issue.

As for our custom project, we decided to implement a photoresistor sensor that allows the quadcopter to take off or land, based on the lighting conditions. We were only able to make the quadcopter fly up and down when the lighting changed; however, with more time, we intend to further develop our prototype photoresistor algorithm to completely land and fly.

This report contains a comprehensive design overview and specifications for each of our prototypes: the maze traversal algorithm and our custom addition. It also details the circuitry and coding algorithms of each project, as well as future improvements and applications of these tasks.

1. Introduction

We experience the shift between day and night every single day of our lives. If we could develop an algorithm capable of capturing this change, our devices and automated systems would be able to function on the same schedule as us. Not only would such a concept reduce the manual input needed to notify our systems of this shift, but it could greatly improve the efficiency of our autonomous systems and change the dynamics of many homes and workplaces. Ariadne de Bothezat, one of Aquall's clients, was interested in developing and marketing autonomous quadcopter platforms, and, conveniently, we have recently been developing algorithms and prototypes that meet Ms. Bothezat's requirements.

Therefore, President Dick asked all development teams to introduce to the client their autonomous quadcopter and customs projects. Our team, Sunset Industries, developed an algorithm that allowed a quadcopter to successfully and rapidly traverse an obstacle course and land in the landing zone while minimizing collisions. Also, we implement a photoresistor sensor that allows the quadcopter to take off or land, based on the lighting conditions.

2. Criteria and Constraints

Sunset Industries was asked to meet several criteria and had to fit into many design constraints throughout both of these projects.

2.1. Maze Criteria and Constraints

Project Criteria: Our Maze Traversal Project had to meet the following requirements:

- Reach the end maze by starting at the beginning and flying to the end
- Land in the designated landing zone
- Minimize collisions
- Maximize speed and smoothness of the system

Project Constraints: It was understood that my team needed to code and test the program entirely by ourselves and that we needed to complete it before the end of our lab time. We also needed to take a video of our quadcopter completing the maze for review.

2.2. Light Sensor System Criteria and Constraints

Design Criteria: For our custom project, we had to choose at least one of the following ideas-

- Add a new sensor or actuator to the platform and interface with it to change behavior during flight.
- Develop more sophisticated algorithms, e.g., allowing the quadcopter to do something other than

traverse a fixed maze.
- Instrument the quadcopter hardware and/or firmware to gather some particularly interesting, perhaps introspective, data.
- Interface the quadcopter to interact with another sub-system, such as a remote control or a smartphone.
- Enable the quadcopter to learn things that automatically improve its performance in maze traversal.

My team chose to do a project that incorporated both a new sensor and a sophisticated algorithm.

Project Constraints: The Photoresistor Project had to be lightweight to not alter the flight of our quadcopter. We also needed it to be able to interface with the BeagleBone Blue, so it could alter its flight as needed. Our maximum budget for the Photoresistor Project was fifty dollars ($50). We only needed to purchase a breadboard and some jumper wires. The following components were provided for us:

- Arduino Uno
- Photoresistor
- Resistors (10k Ohm, 3.3k Ohm)
- 4-pin Male JST connector
- USB-A to USB-B connector

We also were provided with the quadcopter and all its necessary components.

3. Quadcopter Maze Project

Sunset Industries' first task was to design a system that enabled an autonomous quadcopter to navigate through a maze enclosure. Supplied with the quadcopter hardware, our team's focus was on writing code and tuning parameters for quick and smooth completion of the maze.

3.1. System Overview

The following section will provide information regarding both the physical and digital systems involved in the maze project.

3.1.1. Physical Systems:

The primary physical systems involved in the maze traversal project were on the quadcopter. The autonomous quadcopters that our team worked with included more electronics than are found on the commonly used pilot-operated drone. These drones come equipped with sensors and an on-board computer, called the BeagleBone Blue, in addition to the components found on widely used drones such as motors and propellers.

The necessary electronics are located at the center of the quadcopter. The BeagleBone Blue sits on the center of the quadcopter's durable plastic frame. It includes an ARM Cortex-A8 microprocessor, enabling it to run Linux code, an Inertial Measurement Unit, containing an accelerometer, a gyroscope, and a magnetometer, and is capable of outputting Pulse Width Modulating signals for controlling the motors. Above the BeagleBone, as shown in Figure 1, is a plastic enclosure that contains four side-mounted LIDAR sensors and an Arduino microcontroller. The LIDAR sensors measure the distance to the nearest object on each of the four sides of the quadcopter, and the microcontroller sends these values to the BeagleBone. A downward-facing ultrasonic sensor underneath the central frame sends the quadcopter's altitude to the BeagleBone.

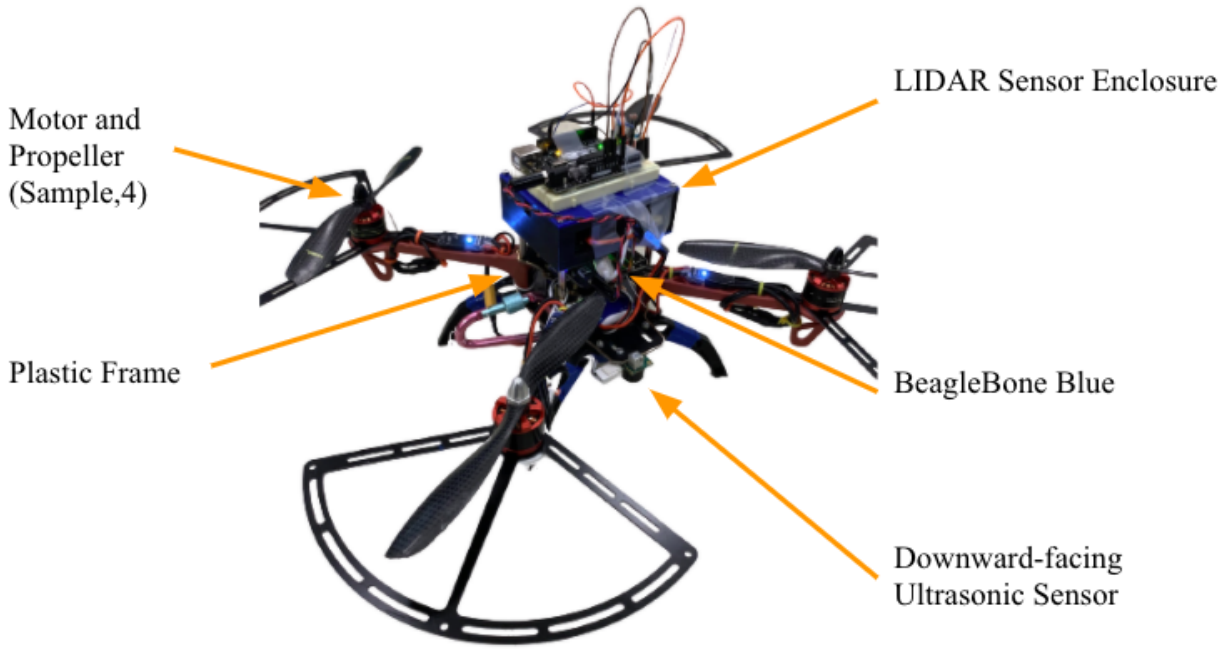

Figure 1: Quadcopter Design Overview

3.1.2. Digital Systems

Our team worked primarily with two different computer programs for this project: Mission Planner and a Linux virtual machine. Mission Planner, as seen in Figure 2, was the program that enabled us to adjust the various flight parameters, tell the quadcopter when to land and take off, and see real-time flight statistics, such as pitch and altitude. The messages feature within Mission Planner was essential for testing and debugging code; it reported which program was running, enabling us to identify why things may be going wrong.

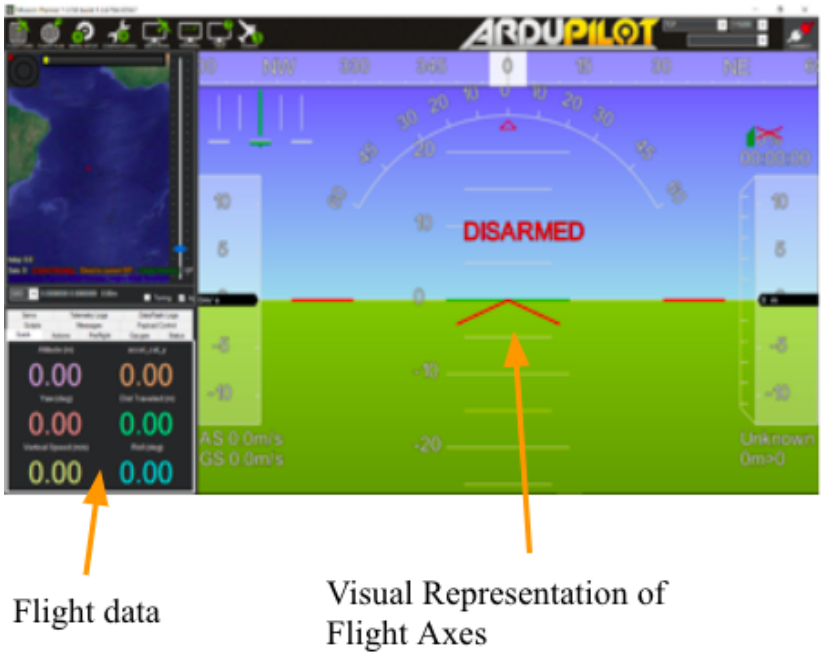

Figure 2: Mission Planner

The Linux virtual machine was where we edited, saved, and uploaded our code for navigating the maze. Given the basic code necessary to takeoff, hold altitude, and land the quadcopter, we were responsible for writing code that would guide the drone through the maze using its sensors.

3.2. Project Specifications

The following section will detail our teams' final parameters and code that led to a successful traversal of the maze.

3.2.1. PID Control and Parameters

For our quadcopters to navigate the maze smoothly and quickly, our team needed to tune various parameters that were used in our quadcopter's code. First, we will discuss PID Control and its related values.

A proportional-integral-derivative controller, commonly referred to as PID, is a control loop system that compares the desired response to feedback from the system to generate its output. An error (e) is obtained by comparing the desired value to the current value as reported by the system. This error is then run through the equation in Figure 3 to determine the desired output (u), where the K values are constants and e(t) represents error as a function of time. PID gets its name from the fact that the function utilizes the proportional, integral, and derivative of the error function. The process of calculating an output based on the current error occurs hundreds of

times per second. The result is a signal output, and hence a physical output (change in altitude, pitch, etc.), that is fast and minimizes overshoot and oscillations.

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

Figure 3: PID Error Calculation Equation

Our autonomous quadcopters can use a PID controller for functions such as altitude and horizontal distance from an object. One of our tasks in this lab was to identify the best constants for ensuring smooth and rapid flight. For context, the proportional (p) term results in the greatest direct change in the output, the integral (i) term accounts for small amounts of error over time, and the derivative (d) term factors in the current rate of change to minimize overshoot. Table 1 summarizes our tuned constant values.

	Roll	Pitch
Proportional Constant	0.9	0.9
Integral Constant	0.3	0.3
Derivative Constant	0.6	0.6

Table 1: Roll and Pitch Parameters

To make the quadcopter's response fast enough to avoid collisions and minimize fluctuations in position, our team opted for high proportional and derivative terms, relative to the integral term. This decision was because a high integral term tends to make the system and its response sluggish.

3.2.2. Flight Code

Once determining the ideal parameters for a smooth and rapid flight through the maze, we began writing the code that our quadcopter would run to determine its route. We opted for a relatively simple algorithm that took advantage of the maze layout, which is illustrated in Figure 4.

Realizing that the quadcopter continuously needed to move to the right when it was able to (assuming starting orientation where facing left in the diagram is forward) and that there was almost always a wall it could use to hold itself centered, we broke the maze into a handful of segments. For example, during the first segment, as seen in the below diagram, the quadcopter needed to hold itself a constant distance from the wall to its left and move forward until it got

close to the wall in front of it. Then it should hold itself a constant distance from that wall and move right until it nears a wall to its right, and so on. The segments we used in our code are shown below.

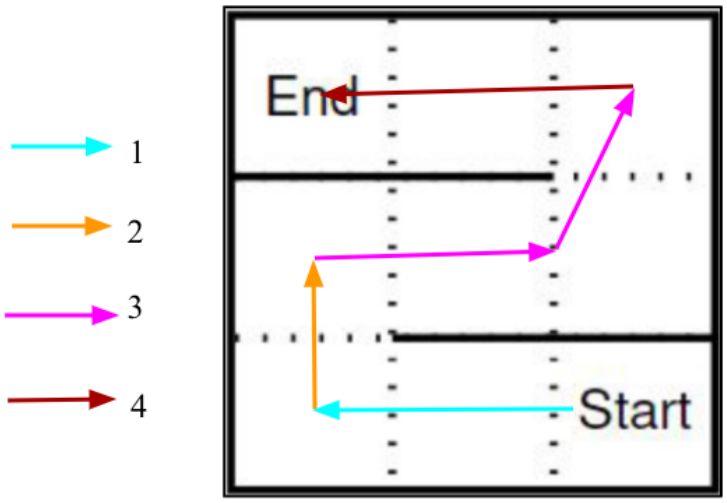

Figure 4: Diagram of the Maze Challenge

Once we wrote the code for the first segment of the maze, the rest was nearly identical. The quadcopter held itself centered in the maze by holding a constant distance of 0.4 meters from one wall and moved in the appropriate direction with a pitch/yaw of 0.75 degrees. Then, once having reached the next segment, it waited two seconds while positioning itself properly from the new and old reference walls. Once the two seconds passed, a counter variable that we used to number each sequence was increased by one, telling the code to run the next section which functioned in an almost identical manner. A sample section code, sequence one, is provided below.

```
//------------------------
// Case 1 : Move forward
//------------------------

if(sequence == 1){
    if(dist_forward > 7){

        target_pitch = -75.0f;
        g.pid_roll.set_input_filter_all(4 - dist_left);
        target_roll = 100.0f * g.pid_roll.get_pid();
    }
    if(dist_forward <= 7){

        g.pid_pitch.set_input_filter_all(10*(0.6f) - dist_forward);
        target_pitch = 100.0f * g.pid_pitch.get_pid();

        g.pid_roll.set_input_filter_all(5 - dist_left);
        target_roll = 100.0f * g.pid_roll.get_pid();

        if(wait_counter++ > 750){
            sequence++;
            wait_counter = 0;
        }
    }
}
```

Figure 5: Sample Sequence Code

During section three of the maze, repeating this exact algorithm was not possible because the reference wall switched sides of the quadcopter before it reached the next wall. Our solution was to hold the reference side the same, even though that wall ended. Due to the geometry of the maze, and as shown in segment three of Figure 4, this meant that the quadcopter completed what would have been the next segment without additional code.

To land the quadcopter, our code returned false once the sequence variable was incremented to five because this meant it had reached the end.

3.3. Results

Our parameters and algorithm for traversing the maze proved successful; our quadcopter reached the end in 44 seconds. The overall flight was smooth and contained few collisions. The biggest issue, which we did not get to address due to time constraints, occurred at the end of segment three. The quadcopter appeared to get "stuck" in this corner and bumped into the wall for a few seconds before finally progressing to complete the maze. Potential future developments and error analysis will be provided in section five of the report.

4. Additional Functionality and Custom Project

In our assignment specifications, our team, Sunset Industries, was also asked to create a team-designed addition to the quadcopter. We decided to add a photoresistor that would measure the light intensity of the room and change the altitude of the quadcopter based on the values it receives. The complete project details are discussed in the following sections.

4.1. System Overview

The team-designed project functions primarily by utilizing a photoresistor that is capable of sensing the light intensity of its environment. As shown in Figure 6 below, the photoresistor is placed on a custom-made circuit board atop the original quadcopter. The quadcopter's battery powers both the BeagleBone blue and the project's Arduino Uno. This Arduino is not connected directly to the BeagleBone Blue but rather to the voltage splitter because the BeagleBone Blue takes in 3.3V but the Arduino Uno sends 5V. A voltage divider is a simple circuit made out of resistors capable of changing an input voltage to the desired value.

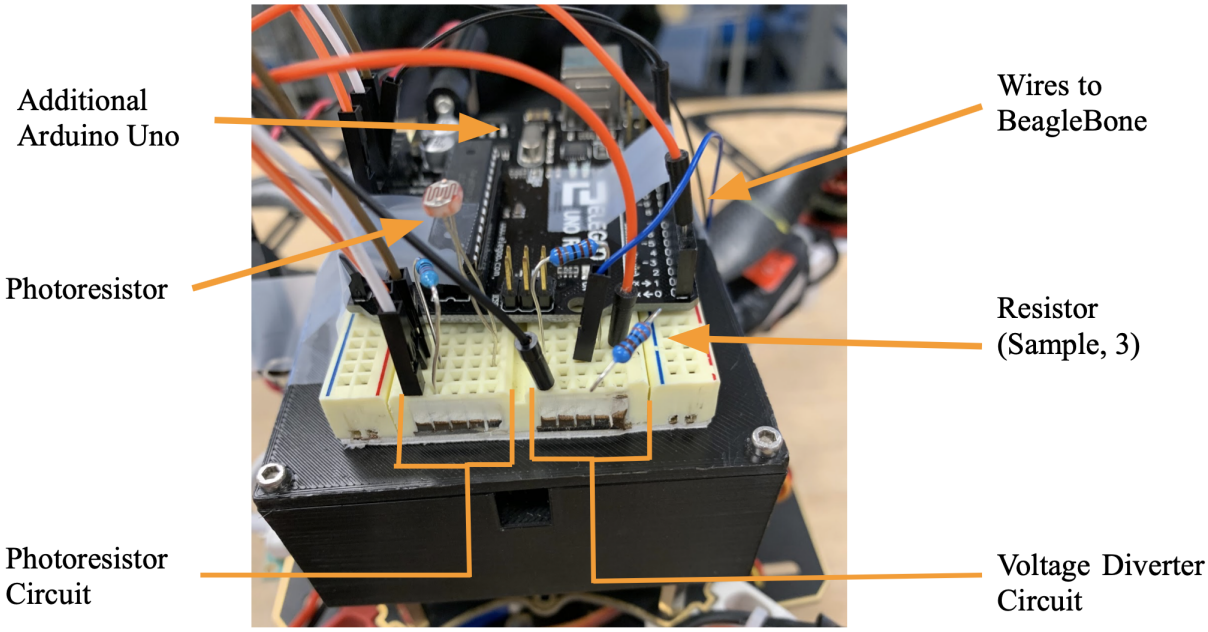

Figure 6: Custom Project Overview Design

The Arduino Uno sits on top of the large breadboard which is secured to the preinstalled Lidar sensor with tape. We chose to use tape due to time constraints and ease of access. The photoresistor sensor was then placed on one side of the breadboard along with the voltage divider. The wires, resistors, and jumper cables were all shortened and secured to ensure the blades of the quadcopter do not slice them. This keeps the quadcopter from malfunctioning after receiving possible extraneous values from the Arduino Uno or photoresistor.

4.2. *Circuitry*

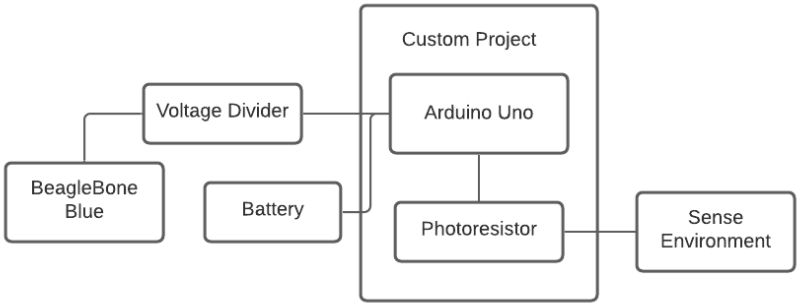

Figure 7: Light Sensor System Block Chart

The circuitry for the team-design project involves our Arduino Uno, which reads the values from the photoresistor and sends them to the BeagleBone Blue. However, if the Arduino was

connected directly to the BeagleBone Blue, it would burn one of the serial ports. Therefore, we must create a voltage divider to send 5V from the Arduino Uno and transform it into 3.3V for the BeagleBone Blue. This is shown in Figure 8 below. Then the 2V lithium-ion battery powers the Arduino Uno which receives data from the photoresistor.

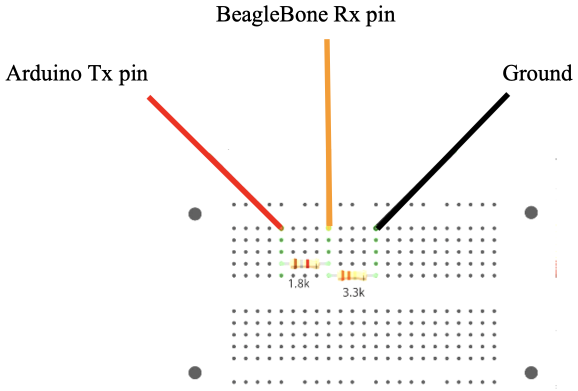

Figure 8: Voltage Splitter Circuit

4.3. *Control Algorithm*

For the project to work as anticipated, we were required to do some coding for both the Arduino to correctly send data to the Beaglebone Blue and for the Beaglebone Blue to use it accordingly.

Receives data

Prints information

Moves quadcopter downwards when lights are off

Moves quadcopter upwards when lights are on

```
// Get downward facing sensor reading in meters
float rangefinder_alt = (float)rangefinder_state.alt_cm / 100.0f;

static int counter = 0;
float pr;
g2.proximity.get_upward_distance(pr);

if (counter++ > 200) {
    gcs_send_text_fmt(MAV_SEVERITY_INFO, "Value: %f", pr);
    gcs_send_text_fmt(MAV_SEVERITY_INFO, "Alltitude: %f", rangefinder_alt);
}

if(pr <= 5.0){
    if(rangefinder_alt >= 0.3){
        target_climb_rate = -7.5f;
    }
    else{
        target_climb_rate = 0.0f;
    }
}
if(pr >= 5.0){
    if(rangefinder_alt <= 0.7){
        target_climb_rate = 7.5f;
    }
    else{
        target_climb_rate = 0.0f;
    }
}

return true;
```

Figure 9: Arduino Uno for Photoresistor Code

Firstly, we have the quadcopter read the data from the ultrasonic sensor and return that value. We then divide it by 100 because we want the value in meters. Next, we create a counter and then read the data from the photoresistor. Then we print both the altitude and the photoresistor value.

The next block of code decides whether to move our quadcopter upwards, downwards, or not at all. We chose a light value of 5.0 to differentiate between the lights being on and off. This value was determined when initially testing the functionality of the photoresistor in different light conditions. The values of the photoresistor ranged from 0 (completely dark) to 10 (completely light).

Then there are the two IF statements towards the bottom of the code. If the lights are off, the quadcopter checks its altitude. If the quadcopter is higher than 0.3 meters above the ground, it will descend. Otherwise, the quadcopter will hold its altitude and hover. If the lights are on, the quadcopter will run a code similar to the previous IF statement. If the altitude is less than 0.7 meters, the quadcopter will ascend, but if it is already at 0.7 meters, it will hold altitude.

5. Future Improvements

Due to the time constraints of our projects, we were not able to complete all of our objectives. Therefore, if we were able to continue the research and development of our systems, there would be plenty of room for improvement.

5.1. Maze Improvements

Due to the limited time we had to work on the maze, many improvements could be made to our system. We managed to complete all of the objectives for this project, however, not as efficiently or effectively as possible. Towards the end of our flight in the maze, our quadcopter was having some difficulty stabilizing and moving forward.

5.2. Light Sensor System Improvements

Our light sensor system was almost successful. As we have stated, we ensured the quadcopter would go up and down when the lights were turned on or off. However, our initial plan was to make the quadcopter fully takeoff and land-based on the light level. We would like to complete this project if we had the time to do so. Furthermore, we would adjust the PID values to create a stable flight during the takeoff and landing.

5.2.1. Future Applications

The situations in which our projects could be used in the real world are numerous. For example, we could use the quadcopter functions to provide fully autonomous flights from different locations around the world. This could save costs on traveling and make traveling more efficient.

As for our custom project, we could expand its light level detection to allow the quadcopter to do some task throughout the day, then when it becomes night, the quadcopter will return to its original position. Overall, this will automize many jobs such as watering plantations and delivering mail.

6. Conclusion

President Robert Dick asked Sunset Industries to develop two main functionalities to implement on an autonomous quadcopter: write code and tune parameters that enable the drone to traverse a maze, and design then implement our custom addition to the quadcopter. In response, we successfully developed an algorithm to complete the obstacle course in a smooth yet rapid manner, and incorporated a new sensor, a photoresistor, onto the quadcopter for our custom project.

Sunset Industries' code algorithm for navigating the maze enabled the quadcopter to reach the end in 44 seconds with minimal collisions. The implementation of a sequential approach to the various segments of the maze allowed for a relatively simple yet effective solution to President Dick's task. By utilizing the constancy of one wall for the quadcopter to use as a reference and a slow but steady pitch and roll to move along, our algorithm proved successful in meeting the requirements set forth for this task.

For the custom project aspect of our task, Sunset Industries decided to implement a light intensity sensor on our quadcopter. We were able to create a successful communication between our sensor, a photoresistor, with the BeagleBone Blue computer via an external Arduino. The extra sensor, along with the new code we wrote and uploaded, added the functionality of making the drone able to adjust its altitude based on whether the lights in the room were on or off. This functionality was able to be fully developed before the deadline set forth by President Dick. Our project only scraped the surface of the massive potential that autonomous systems with integrated day/night detection could have.

In conclusion, autonomous quadcopters have a wide variety of uses and clients. As demonstrated by Sunset Industries' successful completion of the two given assignments, drones have capabilities that could be translated to be relevant for nearly any task or market, regardless of initial constraints in materials or time.

7. Appendices

Appendix A1: Obstacle Course Traversal Code
Appendix A2: Obstacle Course Traversal Video
Appendix B1: Custom Project Code
Appendix B2: Custom Project Flight Video