Project Deliverable H: **Prototype III and Customer Feedback**
GNG 1103 – Engineering Design

Faculty of Engineering – University of Ottawa
Group D2:
Andy Saber
Yusuf Hilal
Steven Huang
Zakkai Shlah
Geoffrey Hooton
March 28th, 2021

Introduction

Over the past couple of months, students from the University of Ottawa's Faculty of Engineering have been working collaboratively with JAMZ delivery to develop various additions to their delivery drones. The group D2 from the engineering design class have decided to help JAMZ by developing an external anti-theft module that's intended to be attached to the outside of the JAMZ drone. In the previous deliverables, the group has developed two prototypes that will serve as a base for this final prototype. The anti-theft module will be controlled by an accelerometer that will detect anything unusual in the drone's motion using the acceleration due to gravity as a reference point. If the accelerometer detects anything unusual, it will change the colour of the lights to red and will activate an automated voice speaker that should inform anyone nearby to stay clear of the drone.

This deliverable will focus on the final prototyping phase of the project. It will contain the specific tests and test objectives to be completed and will show images of the final completed module. Specifically, the speaker and lights will be coded to activate when the accelerometer detects unusual movement, the case will be printed and the Arduino board will be placed in it. We will also clean everything up (mostly the wires) for aesthetic purposes before using it in our final presentation.

Prototype Test Plan

Why are we doing these tests?

What are the specific test objectives?

The first objective for our final prototype is to get our motion sensor properly coded. This means that we are only using the accelerometer to decide if the drone has crashed. The motion sensor registers the altitude change, and once the acceleration reaches the acceleration of gravity and then zero meters per second square, the drone has crashed. Then, the motion sensor will send a signal to the LED strip to turn red. Once the lights have turned red, our speaker system will turn on. This speaker will play a message warning nearby people that this drone is being tracked and that a company representative is on their way to collect the drone. Our next objective is to make sure the Arduino case fits both the Arduino board, and the wires. Finally, we will worry about the aesthetic of the Arduino case, and the organization of the wires.

What exactly is being learnt or communicated with the prototype?

To begin the learning process, we will learn how to connect the speaker to lights in the code. First of all, there is communication between the motion sensors accelerometer and the light system. The lights will turn red once the sensor decides that the drone has crashed. If the drone is in the sky the lights will be green, and otherwise the lights will be yellow or red. Next, the lights will communicate with the speaker. Based on the different coloured lights the speaker will turn on. More specifically, once the lights turn red, a message will play out the speaker. This will be denoted as the “crash state”.

What are the possible types of results?

We will begin discussing if all the results are failures. Our first test is connecting the lights to the motion sensor. We have already done this successfully so there’s no point of talking about failing this. The next results will tell us if we can connect the speaker to the motion sensor. There is a chance that we will not be able to connect the speaker to the motion sensor. The final possible failure would be the Arduino case being too small to fit all the wires.

If we are successful in our testing, the speaker will work with the red lights. The next test that can be a success will be the case. It is possible that the case perfectly fits.

What are the decisions made based on the results?

The first result is about connecting the speaker to the sensor. If we are unable to make this happen, we will make the code for the speaker different. The only other failed test would be the sizing of the Arduino case. If the case is too small, we will have to redesign the case to incorporate the wires sticking straight up.

If the tests are all successful, we will not have to change any of the prototypes, or our planned codes.

What is the prototype and **how** is the test being done?

Results of Previous Prototype

The previous prototypes were very useful in developing this final prototype. Since we were able to code the lights to change colours when needed, we will be able to implement the lights into the final prototype far more easily. Additionally, in the previous

prototype, we were able to set up the accelerometer to detect changes in the Arduino board's movement. This will also help significantly in the implementation of the accelerometer into the final prototype.

Description of Current Prototype

The current prototype is the final product of all of our past prototypes/deliverables. This means that the current prototype is the connection of all of the components tied together into a small, compact module. This prototype has the Arduino board placed into the case. It also has the accelerometer attached to the Arduino board which will be speaking to the LED light and the automated voice speaker which are also attached to the board. When the drone/arduino board is moving as usual, the lights will turn green, if the drone is suspected of being stolen, or if the drone has crashed, the lights will turn red and the automated voice message will activate to deter thieves and possible civilian injuries.

Describe the testing process in enough detail to allow someone to build and test the prototype instead of you.

The first test will be testing the speaker's ability to turn on based on the motion sensor that decided that the drone has crashed. The next testing phase will be testing the intervals in which the speaker plays. We would like the speaker to play a message every thirty seconds, and we will test our code to make that happen. Next, we will be testing the Arduino board fitting in the Arduino case to see if modifications need to be made for the case design. Finally, we will test the LED lights ability to change colours in different situations. The red will turn on after the motion sensor recognizes that the drones acceleration has gone from the acceleration of gravity to zero. The green light will be on if the drone is flying, and the yellow will be on if the drone is being loaded with food. We will decipher "loading the drone" as when the drone engine is off and it is not moving.

What information is being measured?

The motion sensor accuracy is the main value that is to be measured. After completely understanding the motion sensor and its values, the speaker and lights are then connected. The time delay between each speaker message repeat is also measured, as well as the delay between the LED strips changing from green to red.

What is being observed and how is it being recorded?

As soon as the drone crashes, we are observing the red lights and automated voice to activate at the same and right time. For this to happen the motion sensor and accelerometer must have also functioned properly by all the code we implemented in our Arduino UNO. We used the serial monitor on the Arduino IDE in order to

record/monitor the results from the motion sensor. We also used visual and auditory observations to know if the speaker and lights were working how they were supposed to.

What are the required materials and cost?

In the previous deliverables we ordered the majority of the necessary materials for the final prototype. In this deliverable we will only be purchasing our LED light strips for approximately $13. Additionally, we will need the previously purchased automated voice speaker, and accelerometers from the previous deliverables. Finally we will need our 3D printed case.

What work needs to be done?

The project is mostly completed, but some small parts need to be added/ adjusted with our module. First, a “loading state” needs to be added. This is per client request, as they said that yellow lights turning on during loading would be a nice addition. Also, the overall compactness of the module should be worked on. Lastly, we need to adjust the arduino case to be more accommodating to the module, wires, and other parts.

When is the testing happening and how long will it take?

How long is the test estimated to take and what are the dependencies (i.e. what needs to happen before the testing can occur)?

The testing will occur during the week of the March 15th. Our group organized a time to meet up on Sunday, March 21st to collect and combine all of the individual parts of our design. Combining the components and making sure it all works should take about a day. Finally a test will be completed in order to test the functionality of the design. This will only take an hour or two.

Client Feedback

After showing the client our progress, they were generally satisfied. They especially enjoyed the video that we created of the drop test we created to demonstrate the capabilities of our prototype. They commented on the fact that our cardboard box prototype seemed to have many loose wires that we intend to improve on for the final design. This will be fixed by updating our CAD model of the Arduino case to be able to

house all of our components instead of simply the Arduino board. This should definitely improve overall compactness.

Prototype III Compared to Prototype II

Arduino Case:

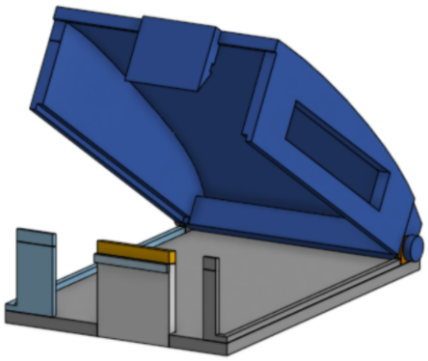

Figure 1: Case design two

Figure 2: Final case design

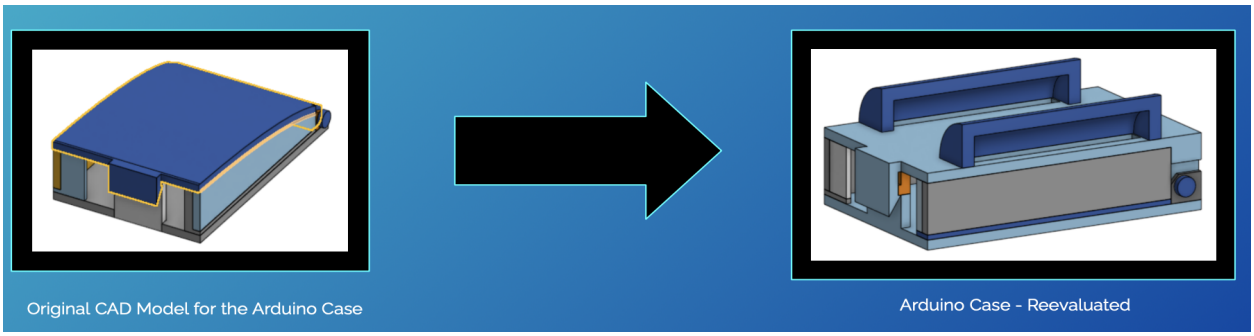

Figure 3: Case design 1 compared to case design 2

On the left is an OnShape model of the Arduino case before it was reworked and printed. It has a sleek aerodynamic design as well as a simple locking mechanism that secured the case. Due to the capabilities of the 3D printer, we could not print the curve on the case; therefore, we had to change the design to a rectangle, removing the aerodynamic design. We also added ventes to the case to allow better wiring of the Arduino.

LED Lights:

Figure 4: Final red LEDs

Figure 5: Final green LEDs

Above are our final pictures of our LED lights activated around our cardboard box which is imitating the drone module. Both the red and green lights are functioning at the proper code implemented as green is for moving normally and red is activated when gravity is -9.8m/s^2 followed by an acceleration of zero.

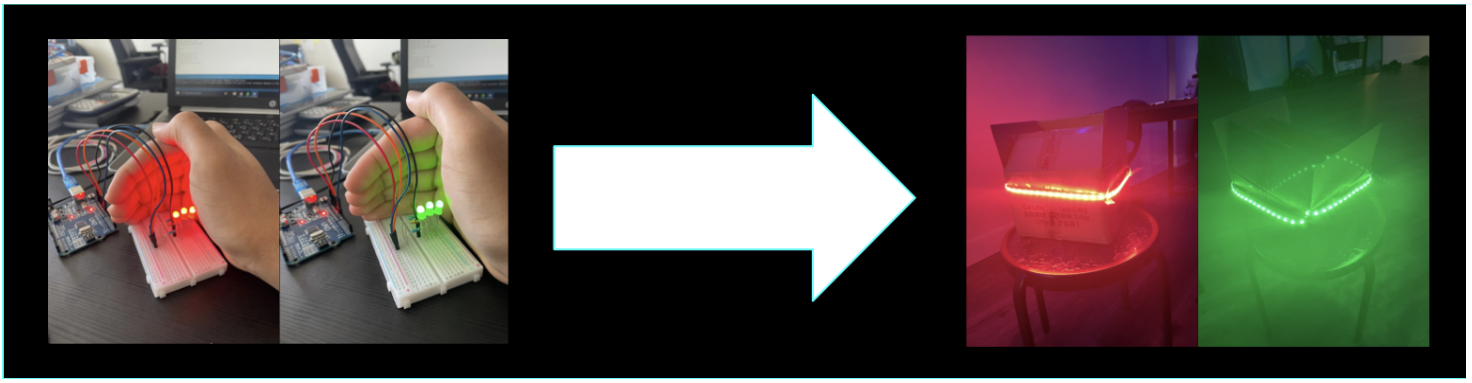

Figure 6: Original LED bulbs compared to LED strips

Above is a picture reference of our previous work for prototype II of our RGB LED bulb lights. We then figured out a better way to show lights by using LED waterproof lights as seen on the right and this is a much more appealing look that we included in our prototype III. We also improved our understanding of voltages and the amount of power needed to power an LED strip.

Prototype III Overall

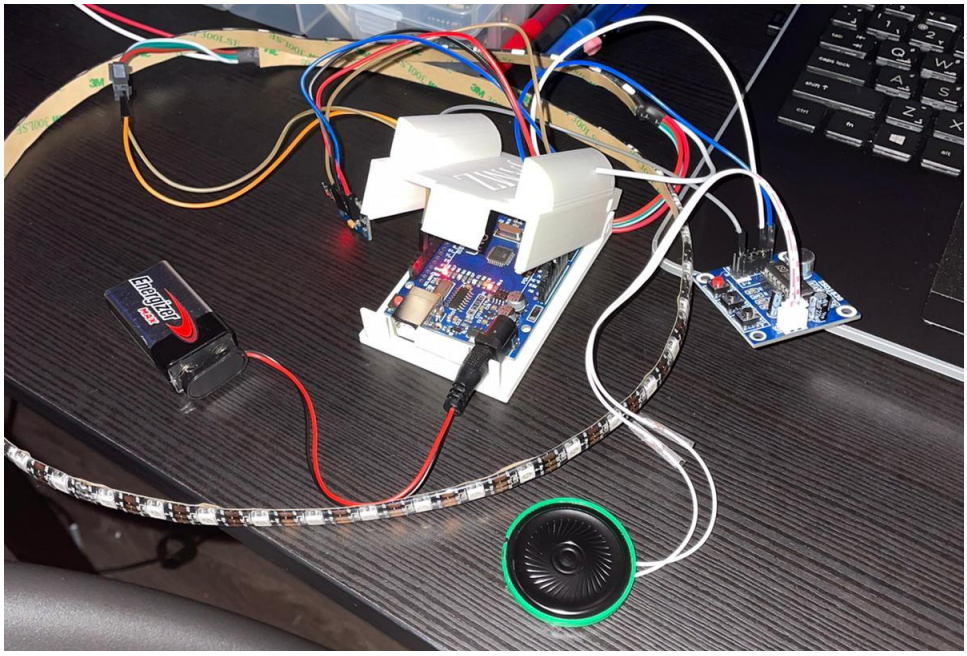

Figure 7: Final prototype of the Arduino board

Above is a picture of the module all connected together. There is the arduino in its case, connected to a motion sensor, a speaker module, and waterproof LED strips. The arduino is powered by a 9V battery, and the LED strip is powered externally (5V). All the components are controlled by the arduino code (*found in the appendix).*

Conclusion:

This deliverable included a summary of the third deliverable created by the group D2 from the Engineering Design class at the University of Ottawa. Our group decided to create an anti-theft module that contains multiple differently coloured lights, and an automated voice, which is all controlled by an accelerometer. In this deliverable, we connected the LED light strips and the automated voice speaker to the Arduino board in order to have them all speaking to each other as intended. We tested these connections by moving the accelerometer rapidly to simulate unusual drone movement. We also tested our 3D printed Arduino case by placing the Arduino board into the case. Finally, our final test included us placing all of the components into a cardboard box and then dropping the box, to make sure that it will be able to simulate a drone crashing.

After this deliverable, we intend to finish our design by adding a yellow LED loading state. Since our Arduino did not fit perfectly in the Arduino case, we will also update our CAD model and re-print the case to make sure that the Arduino board,

accelerometer, and automated voice speaker will all fit into it. This should improve the compactness of the overall module since we had many loose wires in the cardboard box simulation.

Project Planning:

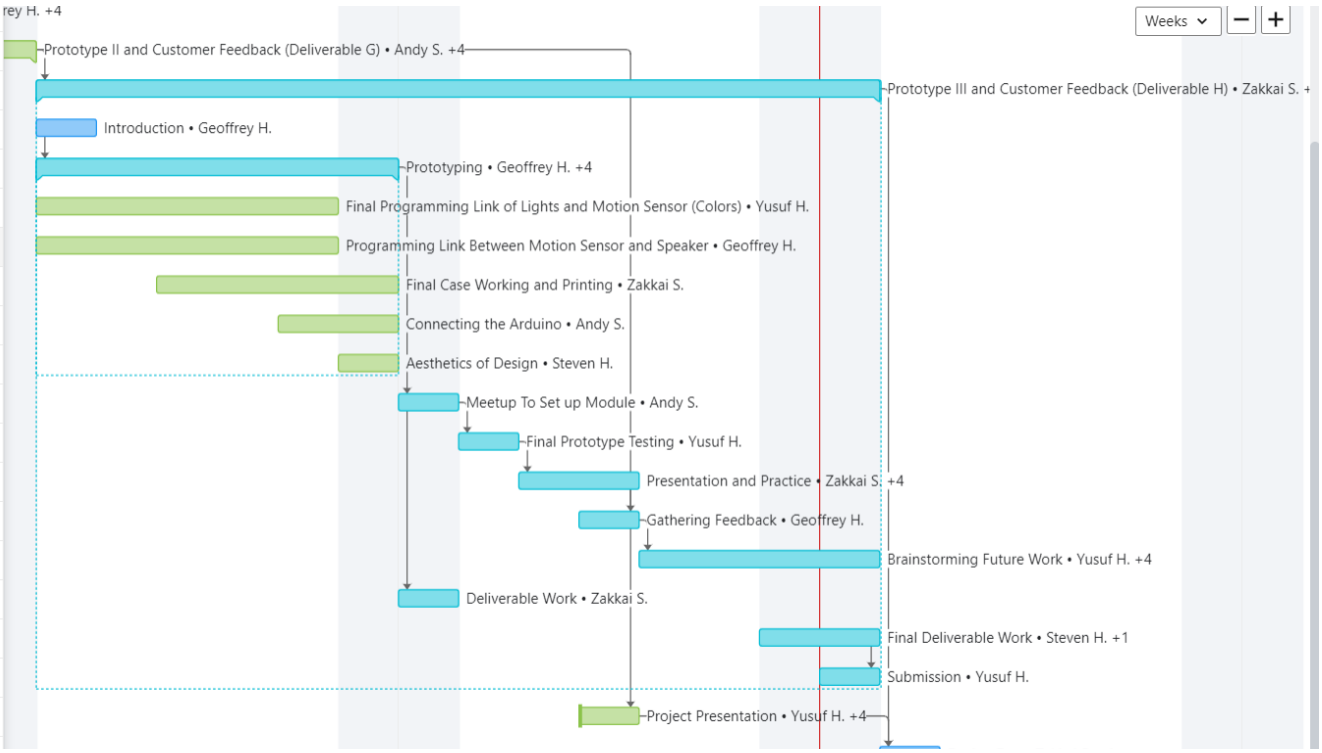

Figure 8: Gantt Chart for the Past 2 Weeks

Figure 8 shows the plan we took to get this deliverable done and get prototype 3 complete. Firstly, it is very important to know that we only work a few hours in the allocated days to avoid stress and overworking. Also, the tasks were divided depending on availability, previous focus, available materials, and strengths. For example, Zakkai was focused on a good chunk of the deliverable work since he couldn't be in Ottawa to work on the physical module. Plus, the dependencies were assigned based on work done and the amount of work left. We therefore guaranteed that no one would be working on 2 things at once unless they were related. Lastly, we also had to take into consideration the presentation and how it will affect and move around our prototyping process. The main effect was that it sped the process up and made us do most of the work in the first week.

The project plan for the next week will depend on work to be done for design day and will most likely follow the same overall ideas to get our work done and hopefully win

design day! The user manual preparation and pitch work will also be included in the plan.

APPENDIX

Arduino Code:

Motion_Sensor_Code | Arduino 1.8.13

File Edit Sketch Tools Help

Upload

Motion_Sensor_Code §

```
#include <Wire.h>
#include <FastLED.h>
#define LED_PIN     7
#define NUM_LEDS    60
CRGB leds[NUM_LEDS];

//initializing variables to use
long accelX, accelY, accelZ;
float gForceX1, gForceY1, gForceZ1, gForceX2, gForceY2, gForceZ2, Xdiff, Ydiff, Zdiff;
int play = 11 ; //speaker pin
int crash = 0; //crash state
int j=0;

void setup() {
  Serial.begin(9600);
  Wire.begin();
  setupMPU();
  FastLED.addLeds<WS2812, LED_PIN, GRB>(leds, NUM_LEDS);
}

void loop() {
  recordAccelRegisters(); //get acceleration values from the motion sensor
  Xdiff = abs(gForceX2-gForceX1);
  Ydiff = abs(gForceY2-gForceY1);
  Zdiff = abs(gForceZ2-gForceZ1);
  for(j; j<1; j++)
  {
    Xdiff = 0;
    Ydiff = 0;
    Zdiff = 0;
  }
  gForceX2=gForceX1;
  gForceY2=gForceY1;
  gForceZ2=gForceZ1;
  printData();
  if(crash == 0) //checking for crash state
  {
    if (Xdiff>=10 || Ydiff>=10 ||Zdiff>=10) //spike in acceleration values
    {
      digitalWrite(play, HIGH);
      digitalWrite(play, LOW);
```

```
      for (int i = 0; i <= 59; i++)
      {
        leds[i] = CRGB ( 255, 0, 0);
        FastLED.show();
        delay(5);
      }
      Serial.println("Red ON");
      crash = 1; //Crash state ON
      }
    else
    {
      for (int i = 59; i >= 0; i--)
      {
        leds[i] = CRGB ( 0, 255, 0); //Green ON
        FastLED.show();
      }
      Serial.println("Green ON");
    }
    }
    if (crash ==1) //used to loop crash state
    {
      digitalWrite(play, HIGH);
      digitalWrite(play, LOW);
      for (int i = 0; i <= 59; i++)
      {
        leds[i] = CRGB ( 255, 0, 0); //Red ON
        FastLED.show();
      }
      for (int i = 0; i <= 59; i++)
      {
        leds[i] = CRGB ( 0, 0, 0);
        FastLED.show();
      }
      Serial.println("Red ON");
    }
  delay(500);
}

//setting up the motion sensor
void setupMPU(){
  Wire.beginTransmission(0b1101000); //This is the I2C address of the MPU
  Wire.write(0x6B); //Accessing the register 6B - Power Management
  Wire.write(0b00000000); //Setting SLEEP register to 0.
  Wire.endTransmission();
  Wire.beginTransmission(0b1101000); //I2C address of the MPU
  Wire.write(0x1C); //Accessing the register 1C - Acccelerometer Configuration
  Wire.write(0b00000000); //Setting the accel to +/- 2g
  Wire.endTransmission();
}

```

```
//getting acceleration from the motion sensor
void recordAccelRegisters() {
  Wire.beginTransmission(0b1101000); //I2C address of the MPU
  Wire.write(0x3B); //Starting register for Accel Readings
  Wire.endTransmission();
  Wire.requestFrom(0b1101000,6); //Request Accel Registers (3B - 40)
  while(Wire.available() < 6);
  accelX = Wire.read()<<8|Wire.read(); //Store first two bytes into accelX
  accelY = Wire.read()<<8|Wire.read(); //Store middle two bytes into accelY
  accelZ = Wire.read()<<8|Wire.read(); //Store last two bytes into accelZ
  processAccelData();
}

//changing acceleration values to readable values (m/s^2)
void processAccelData(){
  gForceX1 = accelX *9.81 / 16384.0;
  gForceY1 = accelY *9.81 / 16384.0;
  gForceZ1 = accelZ *9.81 / 16384.0;
}

//printing data onto the serial monitor
void printData() {
  Serial.print(" Accel (m/s2)");
  Serial.print(" X=");
  Serial.print(gForceX1);
  Serial.print(" Xdiff=");
  Serial.print(Xdiff);
  Serial.print(" Y=");
  Serial.print(gForceY1);
  Serial.print(" Ydiff=");
  Serial.print(Ydiff);
  Serial.print(" Z=");
  Serial.print(gForceZ1);
  Serial.print(" Zdiff=");
  Serial.println(Xdiff);
}
```