GNG1103

# Design Project User and Product Manual

# DRONE ANTI-THEFT MODULE


Submitted by:

The Foreigners Group D2

Andy Saber, 300166437

Geoffrey Hooton, 300187367

Zakkai Shlah, 300152659

Steven Huang, 300076558

Yusuf Hilal, 300202318


April 11th, 2021


University of Ottawa

# Table of Contents

OBJ

# List of Figures

# List of Tables

# List of Acronyms and Glossary

**Table 1. Acronyms**

| Acronym | Definition |
|---|---|
| LED | Light-emitting diode, a type of lightbulb |
| RGB | Red, Green, and Blue, a type of lightbulb that can show all colours. |

**Table 2. Glossary**

| Term | Definition |
|---|---|
| Jerk | Change in acceleration over time. |

# 1 Introduction

During the winter 2021 semester, students in the University of Ottawa's Faculty of Engineering have been working closely with JAMZ delivery, an autonomous drone food delivery that was created during the Covid-19 pandemic. JAMZ was created with the hopes of creating a fast and safe drone delivery system that specifically targets rural areas of Ontario. JAMZ presented the Engineering Design class with many different criteria that the groups could decide to try to meet for their design project. The following prototype is an anti-theft module built to deter the stealing of the JAMZ drone if it has been knocked out of the sky.

Our prototype uses an accelerometer to detect exactly when the drone has fallen out of the sky. If it detects that the drone has been downed, it will change the colour of the attached LED light strips from static green to flashing red. Additionally, an automated voice speaker will activate a recorded message that will warn the passersby to stay clear of the drone and that an operator is on the way. Overall, this should serve as an adequate defense against anybody who may try to steal the JAMZ delivery drone.

This User and Product Manual (UPM) provides the information necessary for the drone operators to effectively use the drone anti-theft module and for prototype documentation. The following user manual will provide an overview of the module and will explain exactly how to replicate the prototypes that we created. It will also discuss how to use/activate the module. Finally, there will be brief instructions on how to perform maintenance on the module.

# 2 Overview

JAMZ requires an external module to be able to recognize if the drone has crashed, so the company knows if the drone is at risk of theft. Additionally, we will notify potential thieves to stay clear of the downed drone in order to ensure human safety.

After our initial meeting, JAMZ gave us a clear list of client needs. From this list, the most important factor is safety. This need influenced us to keep the people on the ground safe, and the drone safe from theft. To do this, we decided, a warning must be given to passers-by to stay away from the crashed drone.

What differentiates us from the rest of the groups is our advanced lights and sound systems. We decided to use LED strips instead of the bulbs for a brighter light warning system. This also allowed us to include different colours with the same light bulbs. The next difference is our automated speaker system. We will have the speaker connected to a motion sensor that will play a message warning people when the drone has crashed.

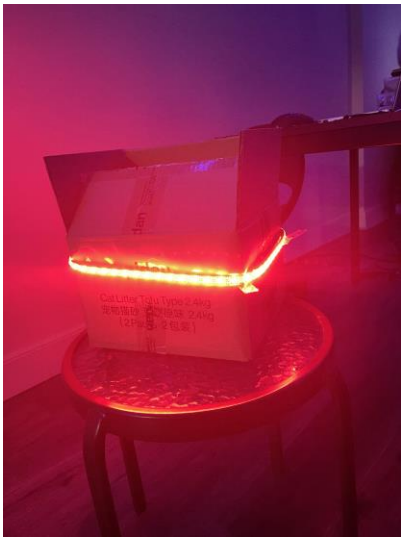

Figure 1: Emergency lights

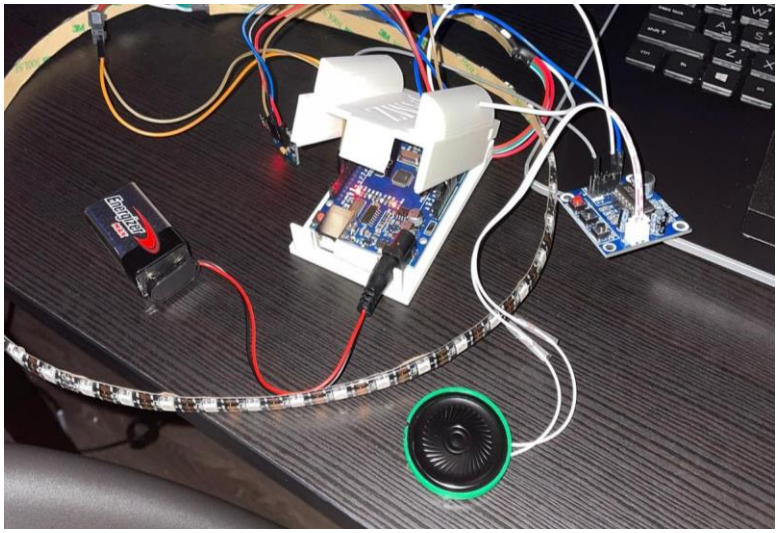

Figure 2: Prototype 3

The key features of our anti-theft module include a speaker with an automated message warning nearby people of the dangers of the downed drone. It also includes bright LED lights to add a visible warning. Our architecture starts from a battery which plugs into the Arduino. The Arduino then controls the motion sensor. From there, the motion sensor controls the lights and speaker.

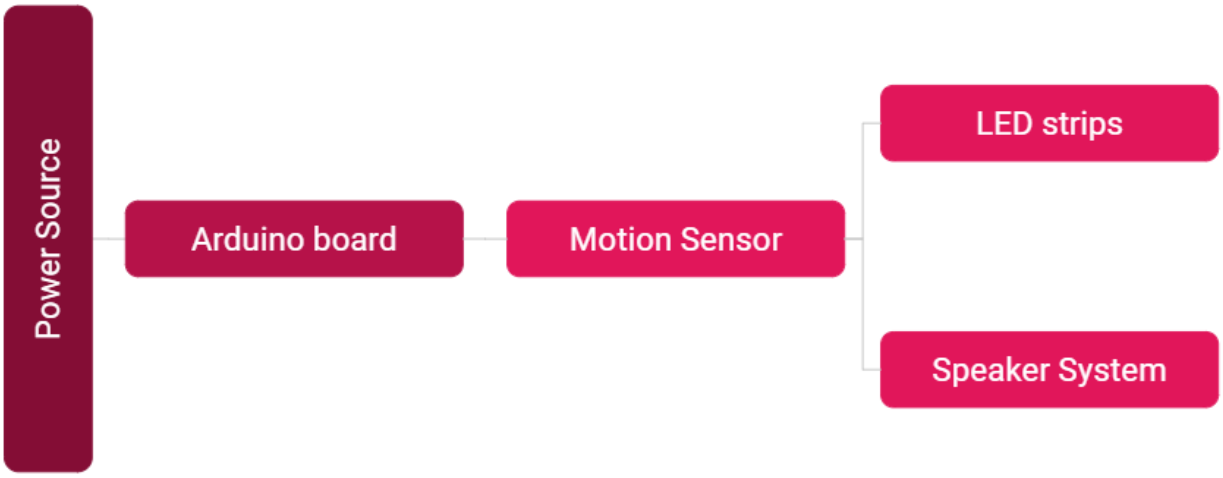


**Figure 3: Block Diagram**

## 2.1 Cautions & Warnings

It should be noted that while the LED light strip is waterproof, the rest of the system is not and should be kept away from water when possible. The wires may also be subject to disconnection, so that must be taken into consideration. Lastly, the power consumption, while not that much, must be regularly checked **safely**. If the drone that the anti-theft module is attached to has undergone any sort of damage, the module should be checked to make sure that all components are working as they need to be.

# 3 Getting started

The system/module is easy to understand. Having 3 subsystems, it is very important to understand each starting with the motion sensor. The motion sensor is coded to get the acceleration of the drone and use that to control the other subsystems. When the motion sensor detects a disturbance in the motion, the rest of the code is triggered. This is known as the "crash" state. The code is very simple and could be adjusted to detect even small disturbances.

The next important subsystem is the lights. They are simple to connect, and they can be controlled to show any color with a red, green, or blue. The main colors used are red and green, but yellow is also implemented into the code. The only issue with the lights is that the LEDs need

an external power source. We decided to use LED light strips instead of normal LEDs so that it would be significantly brighter.

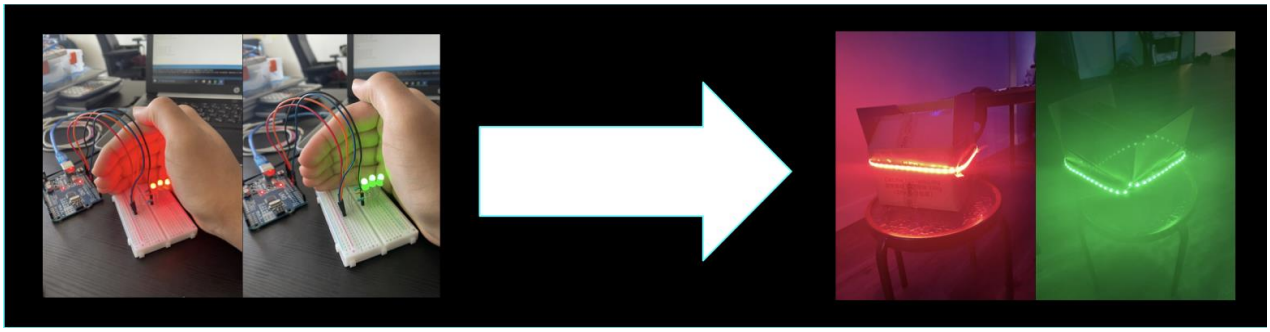

**Figure 4: LED Light Strips as Compared to Regular LEDs**

The speaker subsystem, much like the lights, is easy to use and understand. The automated voice plays as soon as the motion sensor tells it to. The voice is to deter people from getting close to the drone, as well as scare off potential thieves.

Overall, the module is controlled by the motion sensor/accelerometer. It decides on when the drone is entering a different state, such as the crash state. It can tell if the drone is crashing by detecting if the acceleration of the drone is near the acceleration due to gravity $g=9.81 m/s^2$.

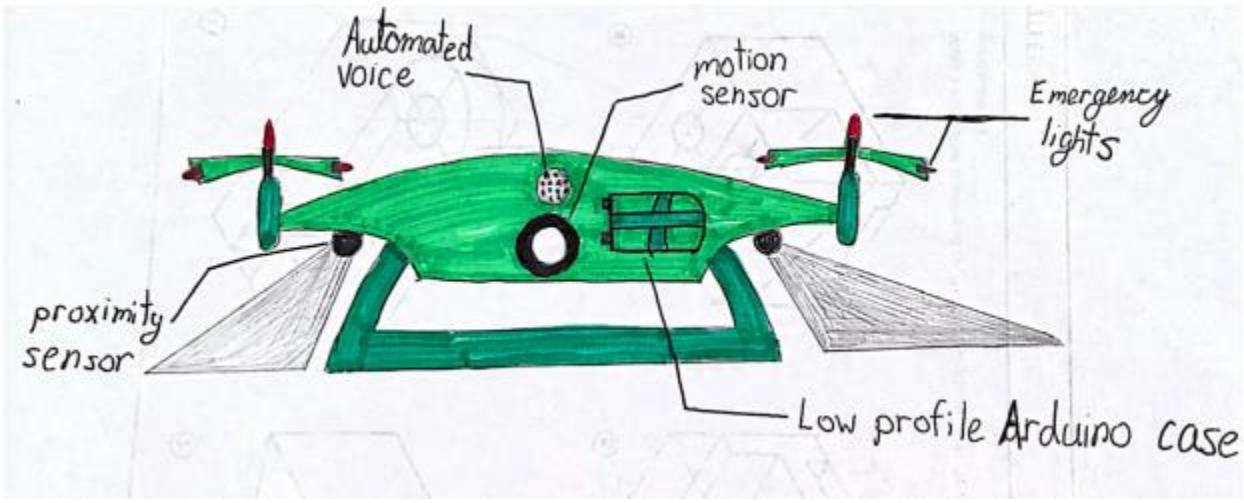


**Figure 5: Overall Concept for Prototype 1**

Specific consideration should be taken into account when connecting the Arduino to the subsystems, and connecting the power source to the LED strip, the Arduino, and connecting the ground wires. See part 6.1.2 for connections.

## 3.1 Set-up Considerations

Our Arduino board communicates with our motion sensor to tell it to turn on the lights and the speaker once there is a large enough change in acceleration over an amount of time. Even though we are using an accelerometer, the motion sensor only activates the emergency anti-theft module once there is a large enough jerk factor (as shown in Figure 4).

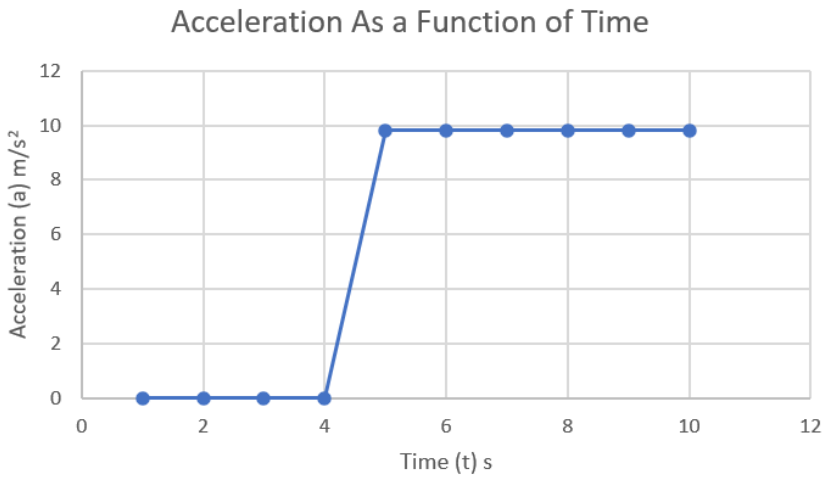


**Figure 6: Jerk Graph**

This graph depicts an estimation of the acceleration that the accelerometer reads in the vertical axis when the drone crashes. The acceleration would be zero in the vertical direction if the drone is hovering at a constant altitude, but once the engines are no longer keeping the drone at that altitude, the motion sensor will sense the gravitational acceleration of 9.81 meters per second squared. As we can see, there is a large change in the acceleration over a short period of time, and, therefore, a large jerk. This will cause the LED lights and the speaker to turn on the crash state.

## 3.2 User Access Considerations

The main user for the drone is JAMZ, as they will be placing the food orders during the loading stage. Other users are the customers as they will be retrieving the food when the drone is at the correct location for the food drop off there that they ordered. The users who are restricted from the drone are any oncoming civilians as they have no authority in touching this drone as it is for both drone's safety and theirs. The only situations in which JAMZ should be required to access

the module is when turning the system on and when turning the system off. Since there is no external power switch so that someone looking to steal the drone can't simply turn off the alarm, JAMZ will have to physically open the Arduino case to turn off the alarm.

## 3.3 Accessing the System

The system is always on when given power. Also, the loading state that is manually triggered serves as a loading state. For control of the loading state, the user is advised to go over the code and see where the input section is (in the appendix).

The system automatically turns on once the motion sensor is triggered. The motion sensor is triggered by large changes in acceleration over short periods of time. This change is called jerk. Once the jerk sets off the motion sensor, the light system will change from green to red; also, the speaker system will declare a message warning nearby people.

## 3.4 System Organization & Navigation

### 3.4.1 Crash State

Whenever the drone is crashing and hits the floor, as soon as the drone's acceleration is equivalent to the earths gravitational pull, flashing red lights turn on with an automated voice too. This is the anti-theft feature we have implemented as the flashing red lights send a warning signal and the automated voice will clearly tell the oncoming civilians to not touch the drone.

### 3.4.2 Loading State

The loading state is the state where JAMZ will set a condition in our code such that the lights will be turned on as yellow to load items on the drone. Within this state, no voice is being played, it is simply a signal to the workers that the drone is currently being accessed in terms of loading food onto the drone for the customers.

### 3.4.3 Normal State/Flying State

This state is going to be for when the drone is flying in the air to drop off a normal delivery to a client. The LED lights will be constantly bright green in order to show functionality of the drone. No automated voice is being played within this state and the colour should stay green unless it has crashed which would then switch to the crashing state and turn red.

## 3.5 Exiting the System

For safety reasons, our anti-theft module has to be turned off by either setting the drone back into its yellow loading state or by unplugging the battery from the Arduino (in case of emergency). This requires an operator to open the module such that it is not easy for the person stealing the drone from simply turning off the lights and automated voice message.

# 4 Using the System

The module is comprised of multiple subsystems, each relating to the others. The following is a little explanation of the use of the modules, as well as the different settings.

## 4.1 Motion sensor/Accelerometer

The motion sensor/accelerometer allows the operator to see how to drone is moving while its operating including its acceleration and direction. The connection scheme can be seen in section 6.1.3.2. After connecting the motion sensor, the module gets accurate data about the module's acceleration and feeds it to the Arduino in order to decide if the crash state and other subsystems should be activated.

### 4.1.1 Proximity Sensor

Originally, we were going to include a proximity sensor that will sense when people or animals near the drone and start the lights and play the recorded message. However, in a meeting with JAMZ they suggested to leave it out and to have the lights and speaker play constantly instead.

## 4.2 Speaker Module

The speaker will include a recorded message that will be played in loop when the drone is in its crash state to warn nearby people of an emergency. The prerecorded automated message plays in French and English to warn nearby civilians as well as potential thieves. The connections are also found in section 6. The speaker module is also comprised of a playing module and a speaker. An amplifier can also be added to the subsystem to make the speaker louder and more deterring. The message will also be played until a technician comes and disables it or when the battery dies so if it gets stolen the drone will be easier to find.

## 4.3 Arduino Case

The Arduino case allows all the sub systems to be connected and it offers a layer of protection if the drone ever crashes on the ground to protect the components from breaking. We originally planned to have the case on the outside of the drone with an aerodynamic design but later we just ended up deciding to have the case inside the drone to offer more climate resistance.

## 4.4 LED Subsystem

The LED subsystem is also another important part of the module. The LED strip connection, just like the others is found in section 6. The LED code is also included in the appendix. The code controls the colors of the lights, with red for a crash state and green for any other situation. The yellow state is also an option to be manually controlled by the user. The yellow color can also serve as a reset state for the module as well as a loading state for the drone.

# 5 Troubleshooting & Support

We allow for a range in which the acceleration of gravity is accepted. There are also areas of the code which can be edited by JAMZ in situations that require different values for the sensitivity of the motion sensor. Other than the obvious issue that might arise of disconnecting wires, small issues may appear with the module. One of the major points is the sensitivity of the motion sensor. While there is the area in the code, as previously mentioned, the code is currently at a detection of 10 m/s^2 (slightly greater than the acceleration due to gravity). This all ensures that the module can be easily adjusted and maintained.

## 5.1 Error Messages or Behaviors

We have had tests of the motion sensor where if the "drone" does not gain enough velocity while falling, the motion sensor will not sense a crash and does not turn the LED light red and start the recorded message. However, as long as the drone is flying at normal altitudes, the drone will pick up enough speed (in free fall) to create enough jerk for the motion sensor to notice a crash. We have also had issues with the precision of the motion senor. Originally, we planned to have two motion sensors on board to average the values between them; however, due to the budget cut, we could not have two sensors on board.

## 5.2 Maintenance

The wires should be checked on a regular basis to ensure that the anti-theft module is properly connected. Furthermore, the module should be properly tested after each instance in which the drone might be damaged from falling out of the air. This will make sure that the components in the anti-theft module have not been damaged from the fall.

## 5.3 Support

In the case that the user requires emergency assistance/support from our team, they can email Geoffrey Hooton or Yusuf Hilal at their respective emails ghoot072@uottawa.ca and yhila023@uottawa.ca. For safety, in the case that there is some sort of maintenance required, remember to unplug the power cable BEFORE handling the electronics in the module. It would be most beneficial if the support email is formatted as following:
**SUBJECT:** Drone Anti-Theft Module
**PROBLEM:** *Describe the problem in as much detail as possible.*
**NOTES:** *Include any other notes (e.g. context, date that the problem started, etc.) that may be important in solving the problem.*

***Please include a signature indicating exactly who is reaching out for assistance.***

# 6 Product Documentation

## 6.1 Final Prototype

### 6.1.1 BOM (Bill of Materials)

The equipment we used in our project with their corresponding links is as follows:

| ITEM(S) | ESTIMATED COST | LINK |
|---|---|---|
| LED Light Strips | $13 | https://amzn.to/3mqXtPl |
| Male to Female wires | $9 | https://www.amazon.ca/Uxcell-a13040500ux0203-Female-Jumper-Cable/dp/B00D7SDDLU/ref=sr_1_7?crid=Q3IENUO827RS&dchild=1&keywords=female+to+male+jumper+wires&qid=1614366020&sprefix=female+to+male+%2Caps%2C257&sr=8-7 |
| 9V Battery Clip Converter Power Cable | $9 | https://www.amazon.ca/Mr-Power-Battery-Converter-Connector-Guitar/dp/B07FCZZ5JF/ref=pd_vtp_2?pd_rd_w=S2jpM&pf_rd_p=d24b48d8-e9cd-4bb0-98bc-8c16948e998b&pf_rd_r=7X2XG6V5DA0MFBAZ996E&pd_rd_r=d5b98942-551f-4c8d-be0e-6e251dc753b3&pd_rd_wg=G6V1H&pd_rd_i=B07FCZZ5JF&psc=1 |
| Motion sensor | $7 | https://www.amazon.ca/Neuftech-MPU-6050-3-Gyroscope-Accelerometer-Crafting/dp/B00PIMRJX6/ref=sr_1_44?dchild=1&keywords=Arduino+motion+sensor+accelerometer&qid=1615406605&sr=8-44 |
| ISD1820 Sound Voice Recording Playback Module | $10 | https://www.amazon.ca/gp/product/B01N69H0GQ/ref=ox_sc_act_title_1?smid=A3G4BHAI7BWOKN&psc=1 |
| | | |
| Estimated Total with Tax | $44 | |

**Table 3: Bill of Materials**

### 6.1.2 Equipment list

The equipment that is required to build this subsystem is as follows:

- An Arduino Uno board
- Male to female wires and male to male wires
- 9V battery
- 9V battery clip converter power cable
- MPU-6050 Accelerometer Module
- ISD1820 Sound Voice Recording Playback Module
- WS2812B LED Light Strip (~1m in length) (Waterproof)
- External 5V power
- 3D Printer (Free)

### 6.1.3 Instructions

We will now describe instructions for each portion of our anti-theft module below.

### 6.1.3.1 Arduino Board

The following figure is a representation of an Arduino board. It is important to have one for this module to function, as well as to make sure connections aren't all tangled. The Arduino will be stored in the case, and have all the other subsystems connected to the it.

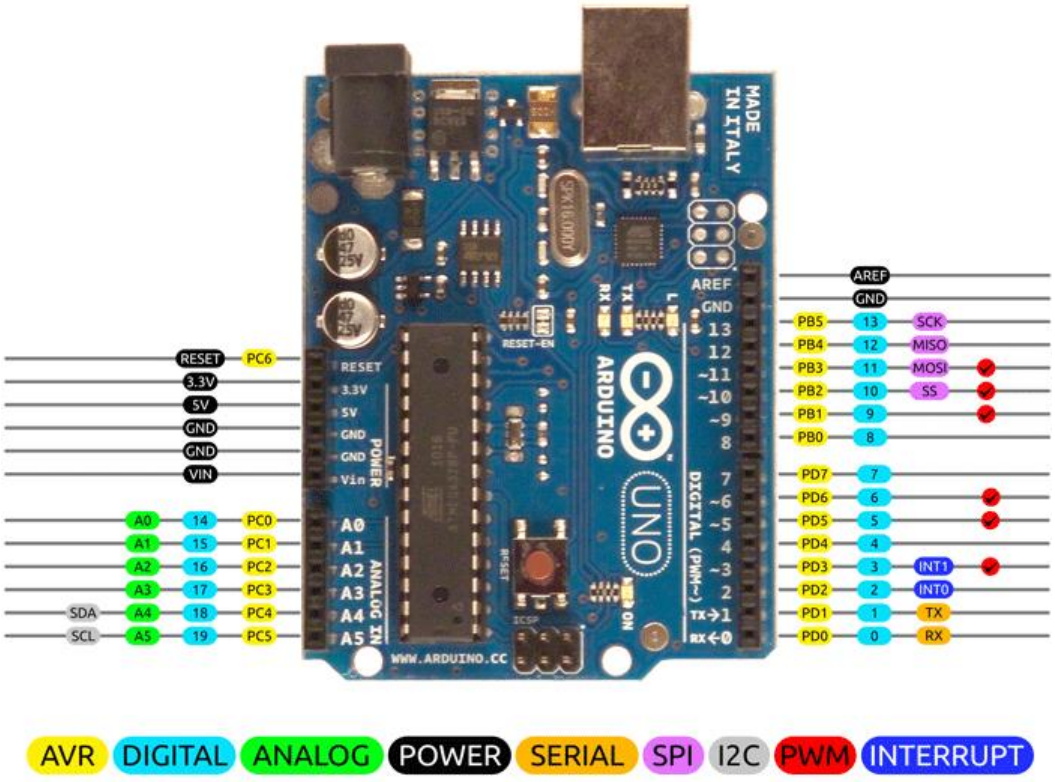


**Figure 7: Arduino board and all connection pin names/functions.**

### 6.1.3.2 Motion Sensor Module

This is arguably the easiest module to connect, as all the pins are connected to the same side of the Arduino. The following figure shows the required connections to have the motion sensor work with the code:

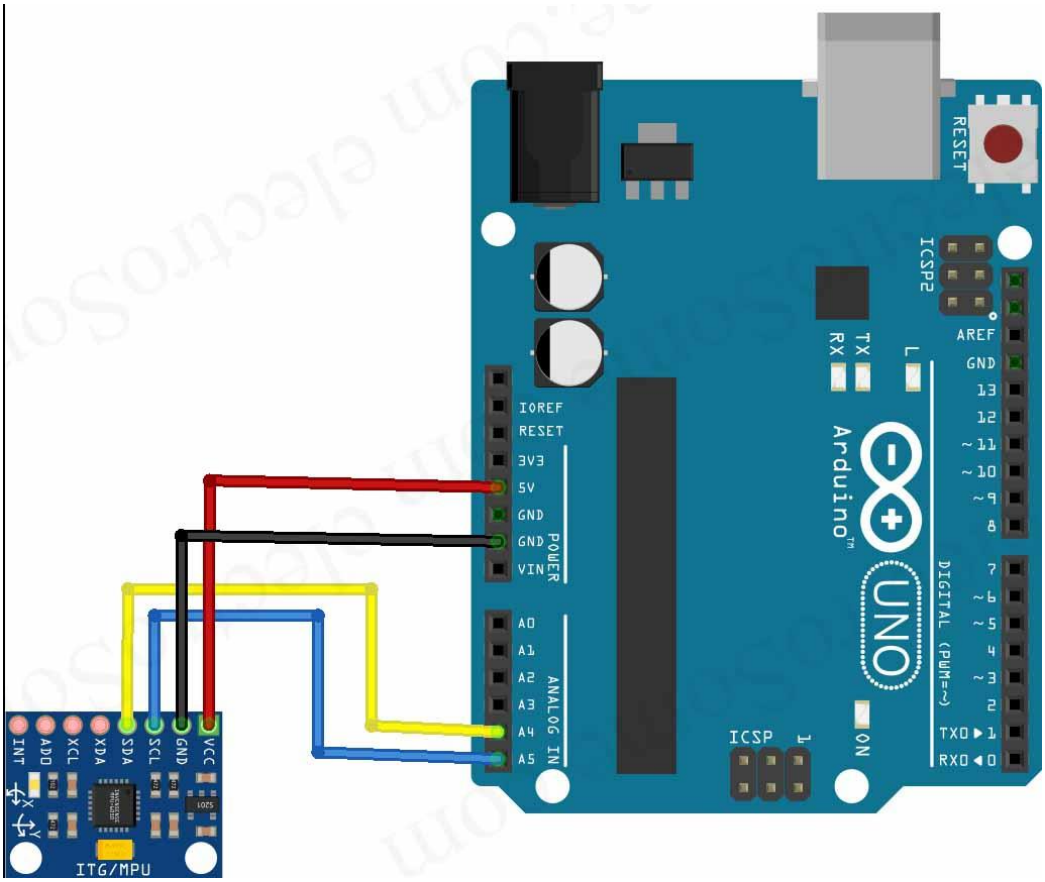


**Figure 8: Motion Sensor Connected to the Arduino Board**

### 6.1.3.3 Arduino Case

To get a three-dimensional printed Arduino case, we first had to design the case in a three-dimensional design software. We decided to use OnShape for this portion of the anti-theft unit. First, we designed the roof of the case and that lid had an aerodynamic curve to it.

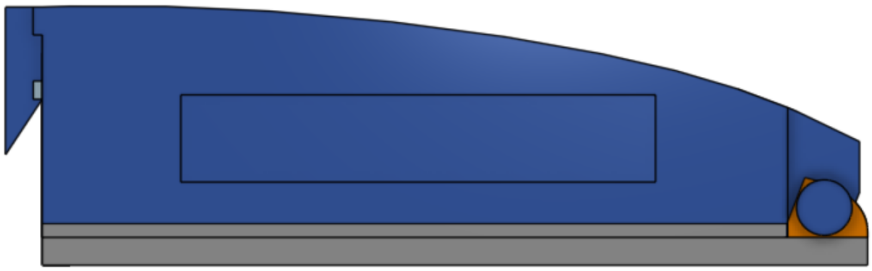

**Figure 9: Original Arduino Case**

However, the 3D printer we were using could not print such a case, so we had to make the case square. Our next step was to make the base of the case. This base had to include a border around the outside of the lid to ensure that the lid would not move side to side. Furthermore, the base of the case required the locking mechanisms. Next, we added water resistant vents at the top of the case to allow for wires to exit out of the case and connect to the LED lights. The final step in making the case was to design the hinge that will allow the case to open and close easily.

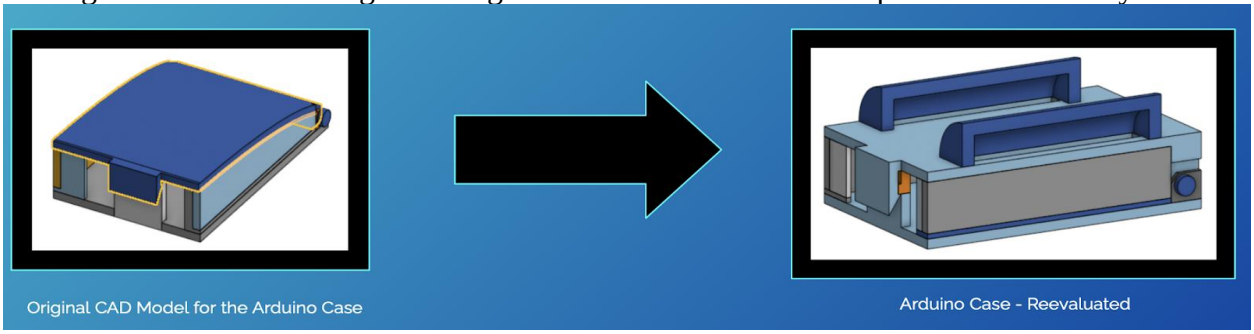


**Figure 10: Original Case versus Final Case**

### 6.1.3.4 Speaker Module

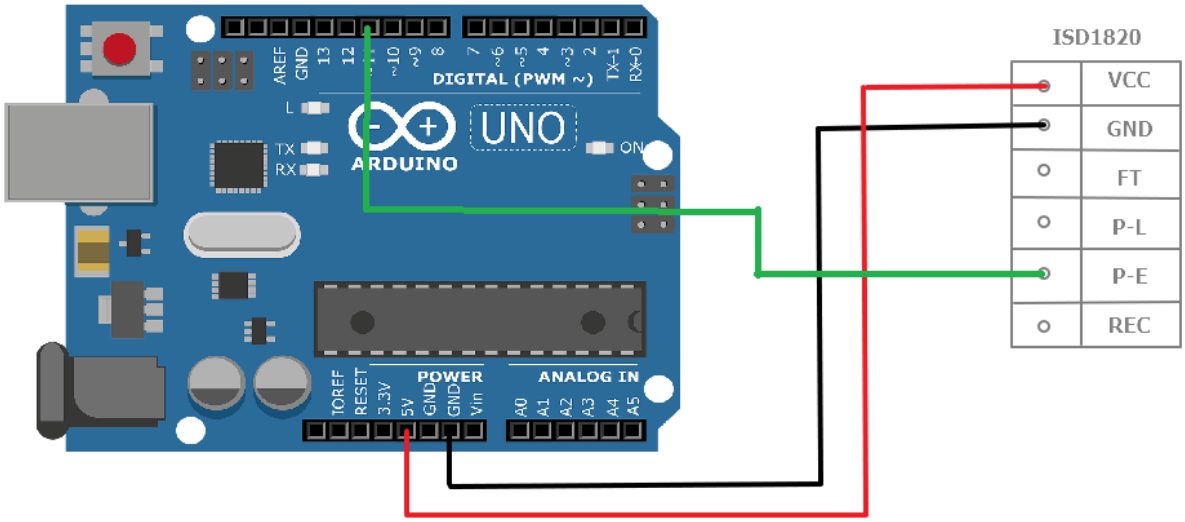


**Figure 11: Arduino Connected to Speaker Module**

The speaker module connection is fairly simple. It requires only 3 wires, with the green one being the one that controls the play or pause setting. The pin that the green wire is connected to on the Arduino depends on the code, as the preset code specifies pin 11, but can be adjusted. Other than that, there are simply some power wires.

#### 6.1.3.5 LED Strip Connection

The LED strip connection is a very essential part of the module and the whole design. It only requires 2 connections to the Arduino. One is the ground wire since it's important to make sure all ground wires are connected through the Arduino for electrical safety. A resistor and capacitor can also be added to save on power and voltage consumption though they aren't essential. The other pin connected to the Arduino is the one that controls the LED strip. The following figure is a good representation of an optimal connection:

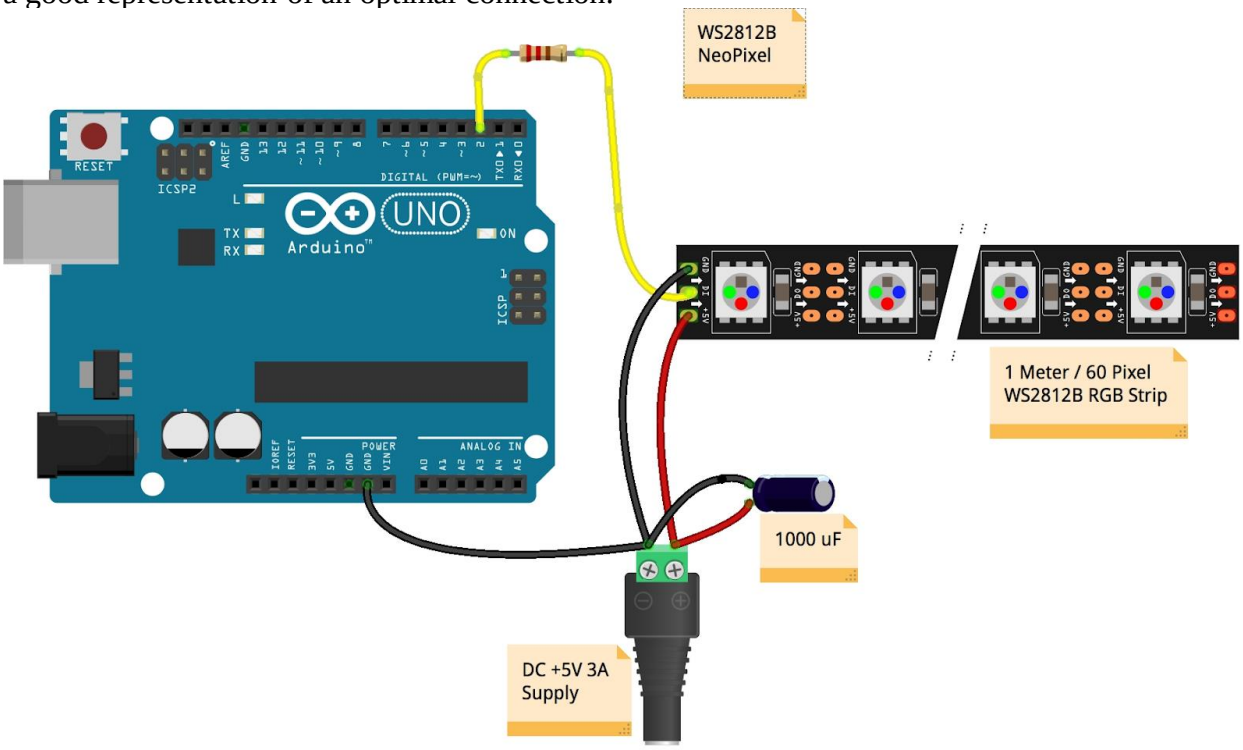


**Figure 12: LEDs Connected to Arduino**

The only 2 issues with the figure are that a) the ground space may be taken, so the other ground (top left of the figure/pin on top of pin 13 in Figure 8) can be used, and b) the pin used by the yellow wire should be connected to pin 7 on the Arduino according to the code. This can also be easily adjusted in the code to accommodate connection to any pin number.

## 6.2 Testing & Validation

***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 light's ability to change colors in different situations. The red will turn on after the motion sensor recognizes that the drone's 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 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.

```
    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);
```

**Figure 13: Light System Code with Loading State Slot**

***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.

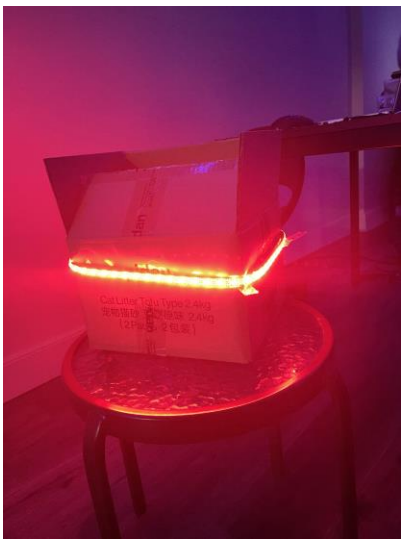

**Figure 14: Final red LEDs**

**Figure 15: 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.

***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.

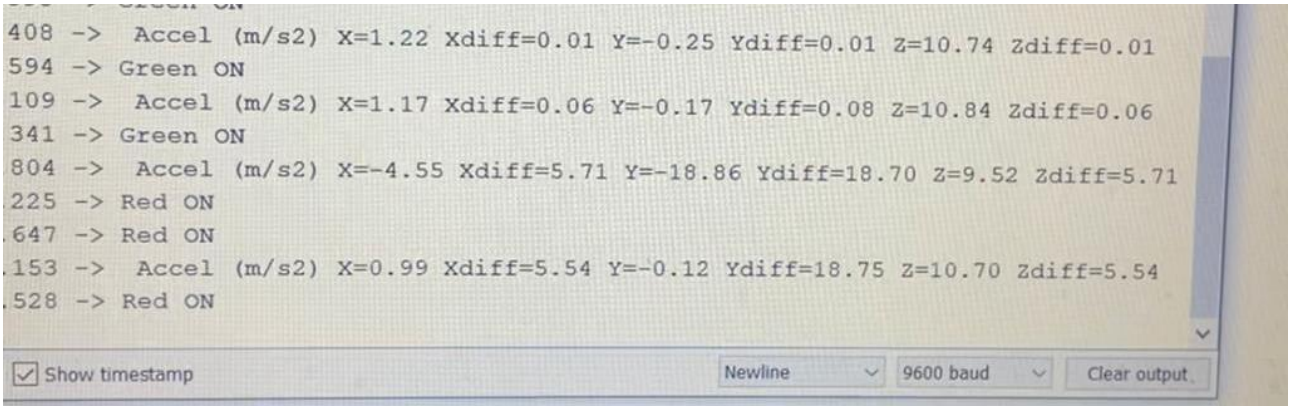


**Figure 16: A large spike in the Accelerometer Data Causes the Crash State**

***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.

# 7 Conclusions and Recommendations for Future Work

## 7.1 Lessons that we learned:

Here are the three biggest takeaways from our project:

1. Manage your time! - This was extremely important with the weekly deliverables and was always especially challenging when working in groups. There would be times we were all overwhelmed and thought we could not make time, but we pushed through it and tried to get work done. While having many other courses this was a big lesson learned and developed our maturity and responsibility skills as well.
2. Expect that roadblocks will appear and prepare to navigate around them! We tried our best to not be discouraged if things did not immediately go as planned, since we needed to learn to work around it. Just because there is a problem in the way does not mean we can stop and complain, it means we need to work even harder for success and keep moving forward, that's how winning is done.
3. Try to get as much client feedback as possible! - as this was often the most substantial and useful feedback that we received during the design process since it defined most of our project. Working with our client JAMZ was outstanding and all of their feedback that they provided us really helped our problems and developed our prototype too. It gave us a

better understanding of what to do and how to attack the problem while making us calmer since we knew there were solutions.

## *7.2* **Future Work**

The project is mostly completed, but some small parts need to be added/ adjusted with our module. First, a "loading state" needs to be worked on/decided on what to be done with from the user's point of view. 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, the Arduino case can be redesigned to be more accommodating to the module, wires, and other parts. By adding all these incorporations to our design this will allow our prototype module to be successful and efficient per client request.

A few things we would have added to our porotype was to include a speaker amplifier. This was in our original prototype idea but due to the budget adjustment, our group could no longer add this feature since it was expensive and did not fit under the $50 budget. Another thing that was due to the lack of time was making our Arduino case a bit larger to fit the motion sensor in it as well. This is something to look forward to if we had a few more months as it will ensure all the anti-theft components are secured together in one 3D-printed case. This will also keep it more organized and appealing.

## 7.3 **Conclusion**

Throughout the 2021 winter semester, students from the University of Ottawa's Faculty of Engineering worked closely with JAMZ drone delivery to help them build multiple modifications for the delivery drones. Our group decided to attempt to build JAMZ an anti-theft module that would discourage anybody that might attempt to steal the downed drone. Our anti-theft module went through many different prototyping phases, and we ended up settling on a module containing LED light strips and an automated voice speaker. These two components would activate when the on-board accelerometer detects that the drone is dropping at an acceleration near the acceleration due to gravity. If it detects this, we programmed the device to turn the LED light strips from green to flashing red, and also turn on an automated voice message. This message would warn (in two languages) anyone around to stay away from the drone and indicate that a JAMZ operator is on the way. We also designed a 3D printed case to house all of the components, although if we were given more time, we would improve the case so that the overall compactness of the module is improved.

Overall, this design project has been very educational and has taught our team many important skills for not only designing a product for a customer, but for working cooperatively in groups. The hands-on experience of working on a product directly with JAMZ, the customer,

was a great way to learn how engineers work in the real world and we have developed many skills that will help us be even better as engineers in the future. Some of the key skills we learned were to listen well to the client, manage our time on the project deliverables, communicate extensively within our team, and overall, how to execute the "design thinking process." Our group had a lot of fun working on this project together and are excited to continue studying the design process in the future.

# 8 Bibliography

*Lime Electric Scooter Rentals*. (2017). LIME. https://www.li.me/electric-scooter

*Matternet Drone Delivery*. (2019, March 26). Matternet Drones. https://mttr.net/

*Zipline - Vital, On-Demand Delivery for the World*. (2014). FlyZipline. https://flyzipline.com/

# APPENDICES

# 9 APPENDIX I: Design Files

MakerRepo Link: https://makerepo.com/yhilal02/821.gng1103d2drone-antitheft

**Table 4: Referenced Documents**

| Document Name | Document Link | Issuance Date |
|---|---|---|
| Deliverable A: Team Contract | https://docs.google.com/document/d/1VgVFz4xhdzQj57ud-B37G5RBcQ_MtTfciojU8KU2IWo/edit?usp=sharing | Jan 17, 2021 |
| Deliverable B: Needs Identification and Problem Statement | https://docs.google.com/document/d/128APYD6_9L3Q95wh6mFuslPWizASKPsmCTu379JwuPg/edit?usp=sharing | Jan 24th, 2021 |
| Deliverable C: Design Criteria and Target Specifications | https://docs.google.com/document/d/1g9xrcAFnURXLCKXuRrdcr-jdfc3voalcf-GtB9L_2ME/edit?usp=sharing | Jan 31st, 2021 |
| Deliverable D: Conceptual Design | https://docs.google.com/document/d/14SSIf8XmGtZ1YuD4M7jaDKEs1UfCqsfIf1KUVCzDjWE/edit?usp=sharing | Feb 7th, 2021 |

| | | |
|---|---|---|
| Deliverable E: Project Plan and Cost Estimate | https://docs.google.com/document/d/1Ds_yz0a3EroeR06VOn5U2dFkc9Z9O3rFqE-3mnRQIiU/edit?usp=sharing | Feb 14th, 2021 |
| Deliverable F: Prototype I and Customer Feedback | https://docs.google.com/document/d/14g4ICtMb41rTpkbEkaGSPoiW6DGwo_QJhLb4okw9ON0/edit?usp=sharing | Feb 21st, 2021 |
| Deliverable G: Prototype II and Customer Feedback | https://docs.google.com/document/d/1I20bftOvJ6kkE9j7F8EouAAVHzFuho2XJUftIy9qV8M/edit?usp=sharing | March 2nd, 2021 |
| Deliverable H: Prototype III and Customer Feedback | https://docs.google.com/document/d/1jHBu1Y3vhLcXomsq6vS1z3C2t80h17ZUiTqlX0UJEqY/edit?usp=sharing | March 7th, 2021 |

# 10 APPENDIX II: 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 morion 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);
}
```