User and Product Manual Instructions

This document is a template of a user and product manual. The client may wish to make improvements on the prototype or need to fix it if something goes wrong or another group of students may work to make a more rugged prototype. The document needs to be clear for someone else who is not an engineer **to use, maintain or reproduce the project**. Include as many images and diagrams for a better understanding. Keep it plain, simple, visual and logical.

In general, if you are not sure exactly what to include, imagine that this document was the only thing that you had. Imagine also that your job was to add a new feature to the project that is described in your document. What would you need to know?

GNG1103

Design Project User and Product Manual

Violent Shaking Detection System

Submitted by:

"Diamond Hands" - Group 6

Group Members	Student Number
Karsten Lowe	300141177
Jason Clapiz	300172134
Leo Tan	300018447
Connor Harper	300166870

April 11, 2021

University of Ottawa

Table of Contents

List of Figures

List of Tables

List of Acronyms and Glossary

Table 1. Acronyms

Acronym	Definition
BOM	Bill of Materials
LC	Logic Convertor
NB	note?
OS	Operating System
UPM	User and Product Manual
VSDS	Violent Shake Detection System

Table 2. Glossary

Term	Acronym	Definition
Comma-separated values	csv	A type of file which allows data to be stored in a tabular format. This form of data can be opened with most spreadsheet programs.
Universal asynchronous receiver-transmitter	UART	A very simple form of device to device communication
General-purpose input/output	GPIO	A 40 pin header found on all current Raspberry Pi boards
Stereolithography	stl	A file format native to the stereolithography CAD software. Often used for rapid prototyping, 3D printing, and computer-aided manufacturing
Inno Setup script	INO	Is a software program for use with Arduino systems. Contains source

		code written in the Arduino programming language and is used to control Arduino boards.
integrated development environment	IDE	software for building applications that combines common developer tools into a single graphical user interface

1 Introduction

This User and Product Manual (UPM) provides the information necessary for drone operators, intermediate clients and consumers to use or recreate the Violent Shaking Detection System (VSDS) and for prototype documentation. This document contains information about the general workings of the VSDS as well as how to use the system, how to troubleshoot potential sources of error and instructions to assemble the VSDS. This document also contains information about how to test the VSDS and ensure all components are working as intended.

With this UPM, anyone should be able to recreate the final working product as it was originally made, some understanding of programming and experience with microcontrollers/ integrated circuits are not necessary but will aid in the understanding of the product as well as the ease of use. Some understanding of how technical drawings work will aid in the understanding and recreation of the component housings provided in this document.

Some assumptions made throughout the document is that one attempting to recreate the VSDS should have a soldering iron, solder, 3D printer, flux, microsoft excel, arduino IDE, and the Raspberry Pi has an SD card, uses Raspbian OS and can run Python 3. If access to one or more of these components is not possible, some substitutes may be appropriate and are outlined further in the UPM. It is assumed that 22 gauge wires are used throughout the VSDS.

2 Overview

Along the entire delivery transport chain, there is a consistent need for quality assurance and the ability to monitor the status of the package until delivered to the customer. Drone delivery is emerging recently as an extension of traditional delivery methods and requires a system capable of ensuring product quality to create better value from their service standpoint.

JAMZ drone delivery requested an add-on module with the purpose of detecting violent shaking. This system development followed constraints which were prioritized from JAMZ needs. Those included a wired system, located above the package which transmits a constant data stream relevant to the status of the package. With user experience in mind, an interpreted status in the form of status updates makes it easier for the operator to understand the real time conditions of the package. Furthermore, the flight data is compiled into graphs to better visualize when the package was shaken, for how long the package was shaken, and how severely the package was shaken.

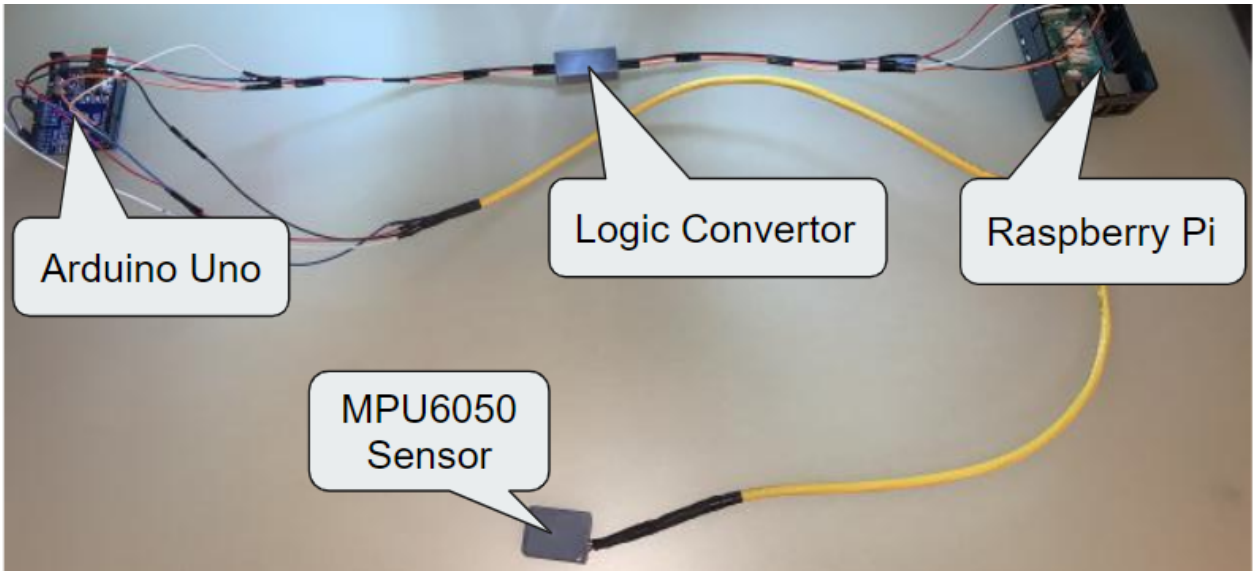

Figure 1. VSDS Final Prototype

The VSDS collects the position data of the package with an MPU 6050 sensor. The data is then manipulated on the Arduino to represent the rate of change of the position data over time. Thresholds are in place to interpret the rate of change, which corresponds to a response of stable, moderate shaking, or violent shaking. The Logic Convertor (LC) reduces the voltage from the Arduino to the Raspberry Pi. The Raspberry Pi displays the real time status of the package in a serial monitor. The flight data is also recorded onto the Raspberry Pi which can be called upon as a graph by the drone operator.

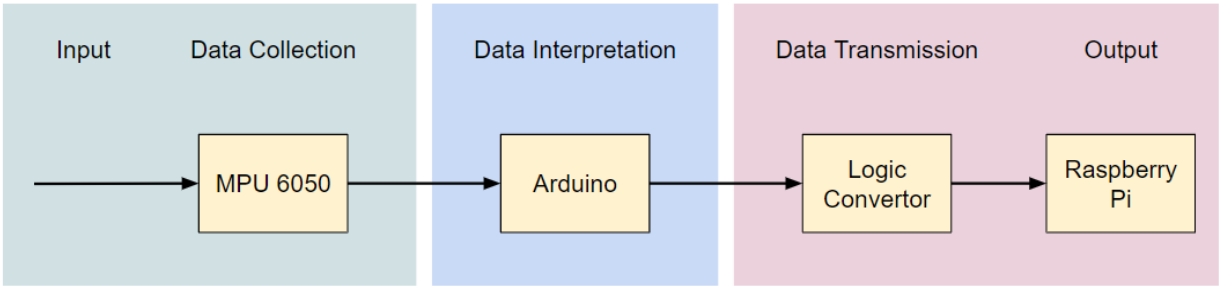

Figure 2. Data Communication Pathway

2.1 Cautions & Warnings

Whenever changing wires on components, or unplugging or plugging in components, ensure the system is not powered and a static shock cannot jump to the components, damage can occur to the components if these processes are not followed.

3 Getting started

The first part of the system is the MPU 6050 6 axis Gyroscope/ Accelerometer. This sensor takes the tilt and vibration the chip feels and converts it into an electrical signal. This signal is communicated to an Arduino microcontroller and is converted into the instantaneous velocity in the pitch, roll and yaw axis of the chip.

The pitch, roll and yaw data is then compared to threshold values determined by arbitrary constants written in the header of the Arduino file. These constants can be changed at any time by changing the numerical variables of the moderateValue and violentValue. A higher numerical value will require more intense shaking to trigger the notification, whereas a smaller value will require less shaking in order to trigger the status.

The data is then printed in the serial monitor as an interpreted status with pitch and roll data values and sent to the Raspberry Pi. In between the Raspberry Pi and the Arduino is a logic converter that steps down the 5v operating signal of the Arduino to the 3.3v used by the Raspberry Pi.

The Raspberry Pi then takes this data from the serial monitor and saves the information to a .csv file, or comma separated value file. This csv file is named after the start date and time of the flight and is stored on the local sd card. This csv file can be opened in Microsoft Excel at a later time and the data converted into a graph using its built in graphing system.

3.1 Set-up Considerations

To recreate this product a few pieces of equipment will be necessary. Soldering iron and solder, 3D printer, flux, Excel, Python 3, and Arduino IDE. If soldering materials are not available, heat shrink connectors and a heat gun or crimping joiners and a crimping tool can be used as a substitute. If access to a 3D printer is not an option, laser cut materials or other hand created casings can be made, however the drawings provided may not be possible to replicate. NB: Using crimping joiners or heat shrink joiners may affect the signal clarity of the final product.

3.2 User Access Considerations

The software has been heavily documented to facilitate understanding of the process behind the code. The code has been developed such that refining the threshold values for moderate and violent shaking are easy and intuitive so that almost no coding experience is required to modify or maintain values. All components have been developed to be hot-swap ready. If any component is damaged or destroyed, switching the part itself with a new copy will allow the system to function as originally intended.

3.3 Accessing the System

Make sure all components are connected and powered on. Next, ensure that the Arduino code is running on the Arduino. Then, run the Python code on the Raspberry Pi. If everything is connected and powered properly with no errors when running the code, you should now be able to access the system.

3.4 System Organization & Navigation

Flight data is recorded as .csv files named by the corresponding date and time, this file is stored inside the user folder on the Raspberry Pi, denoted by the start date and time of the corresponding flight.

3.5 Exiting the System

When you would like to exit the system you will need to stop running the Python code. To do this you will need to enter in a keyboard interrupt. In Python the commonly used keyboard interrupt is ctrl+c, this will interrupt the code and stop it from running. Once the code stops, Python then closes the file you were writing into and saves it. You will have now successfully excited the system.

4 Using the System

The VSDS has 3 features, data collection, interpretation, and saving. As a complete system, the input is registered by moving the MPU 6050 sensor to represent shaking. The VSDS outputs an interpreted status and the … into a csv file.

The following subsections provide detailed, step-by-step instructions on how to use the various functions or features of the violent shaking detection system.

4.1 Data Collection

The data collection system collects the acceleration and gyroscope data of the MPU 6050 in the X, Y, and Z direction to the Arduino where they are calculated to provide instantaneous velocity of pitch, roll, and yaw. These data points are the data found in fractions of seconds and as a result can be extremely high as instantaneous velocity can be upwards of 1000 degrees per second in extreme cases. As the user rotates or shakes the sensor, numerical values should appear in the serial monitor proportional to the intensity of movement. If the sample period of each data point was longer than this prototype, the magnitude of the values should be smaller. This feature does not require any specializations to operate, but an understanding of Arduino based programming is required to understand the calculations that take place to obtain the values of instantaneous velocity or to make modifications to the data sample rate code.

4.2 Data Interpretation

As data is recorded by the Data Collection system, the Arduino interprets the data and classifies it as either “Stable”, “Moderate Shaking”, or “Violent Shaking”. These data values are thresholds that can be modified. These values can be found on line 16 of the Arduino code, the values of 690 for moderate shaking and 820 for violent shaking are arbitrary constants that are compared with inbound values. The higher the number, the more violent the shaking would have to be to trigger the data interpretation of each respective status. As the sensor is shaken, the output of either stable, moderate shaking or violent shaking should be seen in the serial monitor accompanying the recorded data values in pitch and roll.

4.3 Data Saving

The data saving system reads the data that is received from the arduino and creates and writes this data into a .csv file on the Raspberry Pi. The Python code Reads the serial port and receives all the data and outputs it into the serial monitor. The code also simultaneously creates a new .csv file with the exact time and date and writes the received data into this file. The file will keep writing until the code is stopped with a keyboard interrupt. A .csv file is a file that the data is separated by a comma which allows for it to be interpreted by software like Excel and placed into a spreadsheet.

4.3.1 Excel Spreadsheet

The .csv file can be opened using Excel and can be used to graph the data to visually interpret. The .csv file is transferred over to a pc and can be opened using Excel, because the data is separated by commas in this file type, Excel is able to interpret the data into rows and columns placing it into a spreadsheet. Once in a spreadsheet, many different things can be used to interpret this data such as visibly with a line chart.

5 Troubleshooting & Support

5.1 Error Messages or Behaviors

If “nan” appears where numerical data values should be in the serial, this means that the physical connection is interrupted somewhere. Check cables for proper contacts to components, solder components if possible. If wire contacts appear fine, check cables for continuity, it is possible that cables have become damaged internally and the signal is not able to be transferred. Also try resetting the system as sometimes the code gets stuck when wires are disconnected.

If there are issues with permissions to access serial ports on the Raspberry Pi, permissions and/ or port communication may have to be enabled or updated on the Raspberry Pi. A helpful guide to enable port permissions can be found here and a guide to enable hardware permissions can be found here.

5.2 Special Considerations

If using a USB cable rather than a GPIO communication pathway between the Arduino and Raspberry Pi, the python code will have to be modified to accept the USB as a port. Line 7 will have to be changed from /dev/ttyS0 to either /dev/ttyACM# or /dev/ttyAMA# where the # is the number of the COM port used.

5.3 Maintenance

Inspect housings regularly for structural integrity, if plastic is degrading, flaking, or becoming extra brittle, replacement of the case may be required. Check connections between components for corrosion and clean contacts if required. Keep components stored in a cool, dry environment out of direct sunlight. High humidity or direct liquid exposure can harm the components, prolonged direct UV exposure can cause housings to become brittle and break easily. Keep components clean and remove any foreign objects from direct contact with either casings or components. Check wires for damage, if any wires are exposed, fraying or otherwise a fire or shocking hazard, replace immediately.

5.4 Support

Support for any issues can be posted in the comment section of this project's MakerRepo. Problems may already have been addressed and solved, and new questions can be answered by those who have already created and followed the project. A description of the problem should be provided along with any important files or pictures that help demonstrate the issue at hand. Other specific issues can be directed to the Raspberry Pi helpline or the Arduino support page.

6 Product Documentation

The final prototype consists of a power delivery, data gathering, data communication, data storage subsystems and electronic housing. Each subsystem is centralized around the Arduino Uno and branches off from there. For the use of this prototype, 22 gauge wiring is used throughout, as the power draw of components is not particularly demanding. All wire connections that did not plug into the Arduino Uno or the Raspberry Pi, were soldered to ensure a stable connection. If solder is not available, crimping tools, heat shrink wire connectors or male-female wire connectors are all possible substitutes.

The prototype originally created was powered with a 9 volt power supply, it is assumed that most attempts to recreate this prototype will not have a power supply on hand so it is substituted for a 9 volt battery in this document.

The sensor used in the original prototype was an MPU6050. Should other 6 axis gyroscope/ accelerometer sensors be used, code may have to be changed to remain compatible with alternative sensors. For more accurate data collection, a 9 Axis sensor could be used, and would require changes to the provided code to calibrate it.

The Data transmission subsystem uses a GPIO UART connection type, if logic converters are not available, a USB A - USB B connector between the Arduino and Raspberry Pi is sufficient. USB connection is of a lower quality for the purposes of data transmission but is functional.

The data logging subsystem assumes an SD card is already in the Raspberry Pi, other forms of data storage on the Raspberry Pi for the purposes of this prototype have not been tested and could result in varying levels of accuracy and frequency from the original design.

The housings were 3D printed in PLA plastic due to ease of access and quick iteration times, however this stage is optional. If there is not a readily available 3D printer or need for iteration, all housings can be bought from an online retailer. Climate resistance is also necessary for the housings of the system, therefore materials such as MDF would not be recommended.

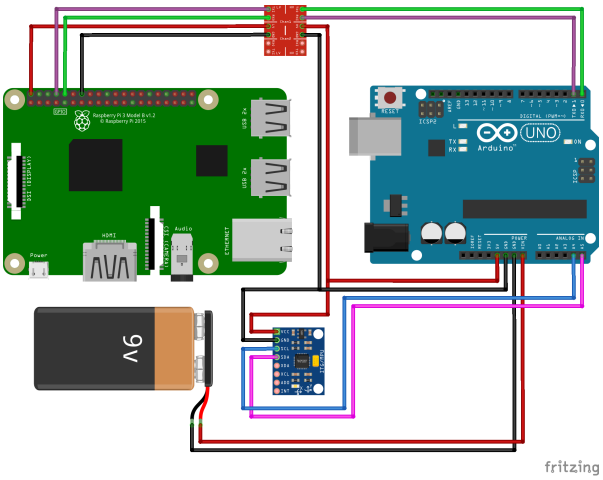

Figure 3. VSDS Circuit Diagram

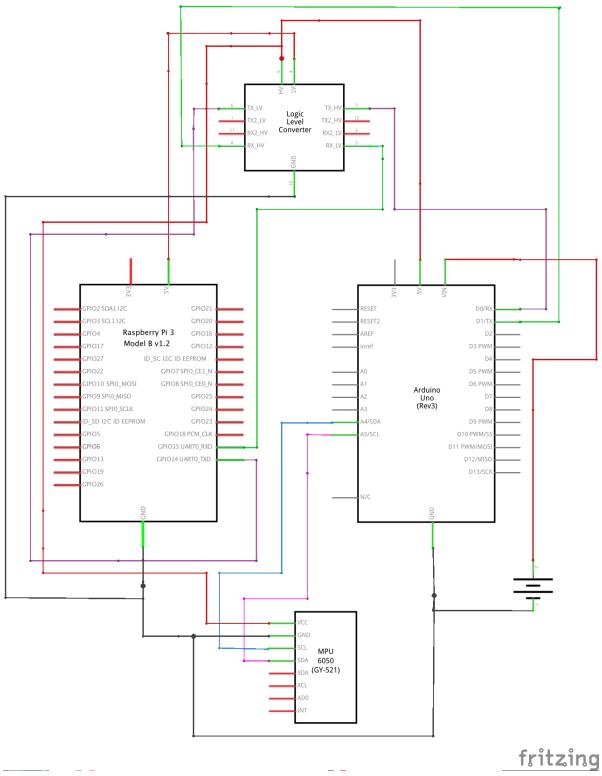

Figure 4. VSDS Schematic

6.1 Subsystem 1 Power Delivery

6.1.1 BOM

Table 3. Bill of Materials for Power Delivery Subsystem

Item	Link	Cost
9 Volt Battery	9V Battery	$3.14/ battery
9 Volt Battery Connector	9V Connector	$2.70/ item
Arduino Microcontroller	Arduino Uno R3	$23

6.1.2 Equipment list

Wire

6.1.3 Instructions

Step 1.

Connect the wires from the Arduino Microcontroller to the 9V Power Supply

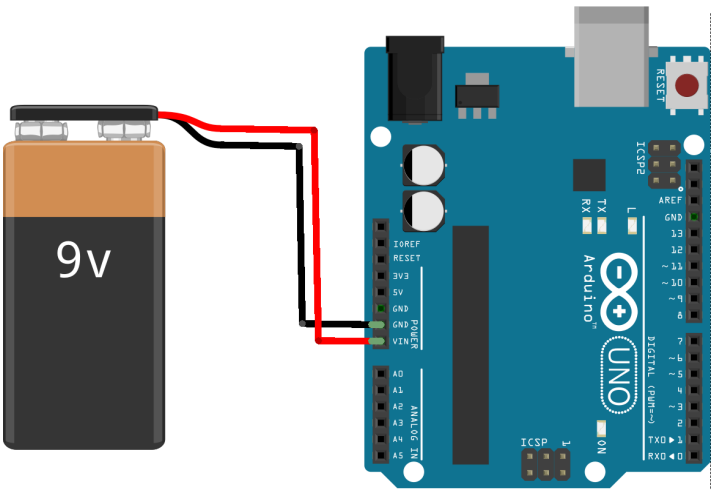

Figure 5. Circuit Diagram 9V to Arduino

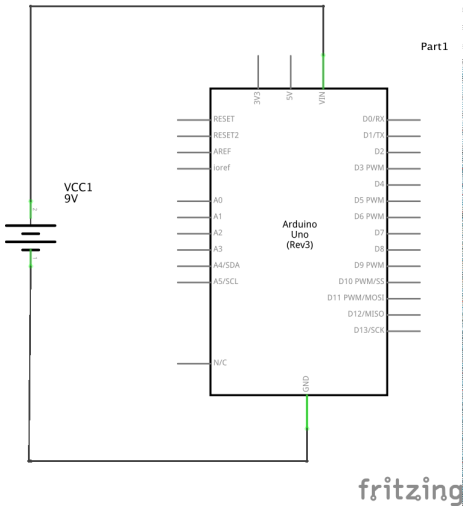

Figure 6. Schematic 9V to Arduino

6.2 Subsystem 2 Data Collection

6.2.1 BOM

Table 4. Bill of Materials for Data Collection Subsystem

Item	Link	Cost
MPU 6050 6 Axis Gyroscope/ Accelerometer	MPU 6050	$3.44
Arduino Microcontroller	Arduino Uno R3	$23

6.2.2 Equipment List

Soldering Iron

Solder

Wire

6.2.3 Instructions

Step 1.

Connect the wires from the Arduino Microcontroller to the MPU 6050.

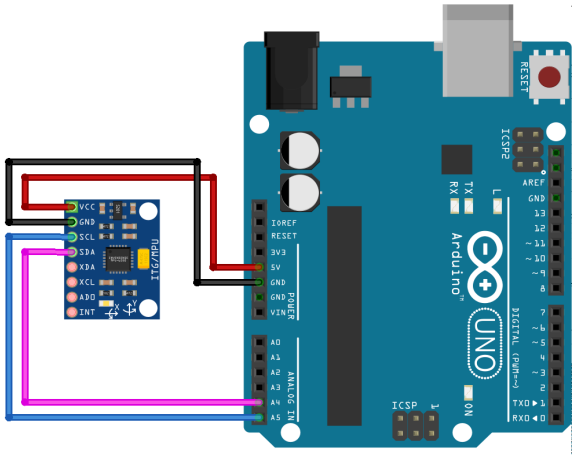

Figure 7. Circuit Diagram MPU 6050 to Arduino

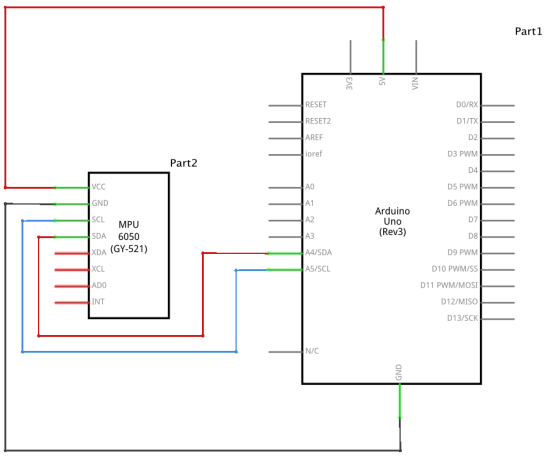

Figure 8. Schematic for wiring MPU 6050 to Arduino

6.3 Subsystem 3 Data Transmission

6.3.1 BOM

Table 5. Bill of Materials for Data transmission

Item	Link	Cost
Raspberry Pi 4	Raspberry Pi	$47.75
Arduino Microcontroller	Arduino Uno R3	$23
Logic Converter	Logic Converter	$9.95

6.3.2 Equipment List

Soldering Iron

Solder

Wire

Latest Arduino IDE (software)

Provided Arduino Code (i.e. ArduinoCode.ino)

Latest Python 3 (software)

Provided Python code (i.e. PythonCode.py)

6.3.3 Instructions

Step 1.

Connect the wires from the Arduino Microcontroller to the Logic Convertor.

Connect the wires from the Raspberry Pi to the Logic Convertor.

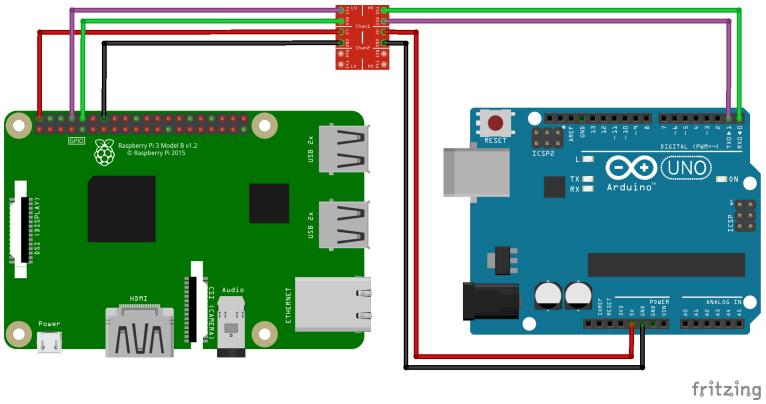

Figure 9. Circuit diagram Arduino-LC-Raspberry Pi

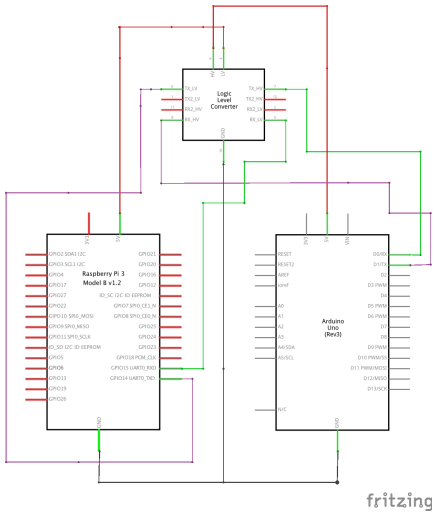

Figure 10. Schematic Arduino-LC-Raspberry P

Step 2.

If you have not already done so, make sure you have enabled UART connections through GPIO on the Raspberry Pi. If you are having any trouble with this a good tutorial is found here. Also ensure all port permissions are enabled. Again, if you are having trouble with this, a helpful guide to enable port permissions can be found here.

Step 3.

If the Raspberry Pi and the Arduino Uno are not already properly connected and turned on, then do that first. If the following github link is available, then head to the link and download the two files (i.e. ArduinoCode.ino and PythonCode.py) from the github repository and open the corresponding file with the corresponding IDE. If the github link is not available, then scroll down to Appendix II and **carefully** copy the corresponding code into the corresponding IDE (i.e. Arduino Code.into into the Arduino IDE and PythonCode.py into the Python IDE). Save the respective files with their respective names on to the desktop of the respective computers.

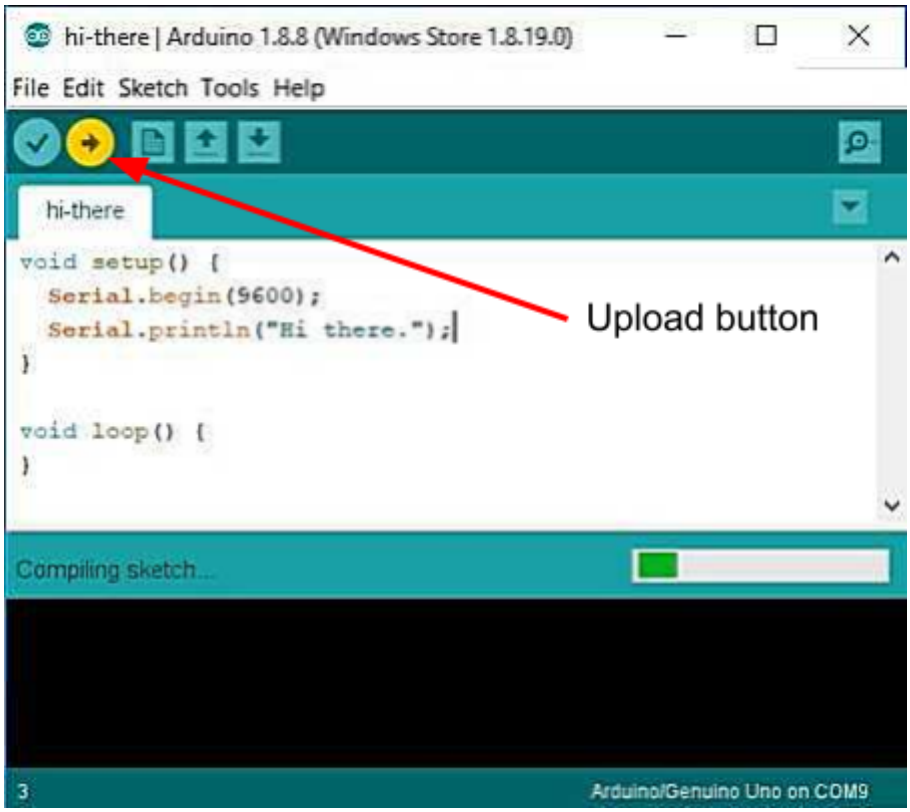

Figure 11. Showcasing location of Arduino upload feature.

Step 5.

In the Arduino IDE, press the *Upload* button to upload the Arduino code that has been copied and pasted or that has been downloaded. The following image shows where the *Upload* button is located, but it is not the real representation of what the Arduino code is supposed to look like. Assuming that the Raspberry Pi has been set up properly both in hardware and software, open up the terminal of the Raspberry Pi and type in the following command without the quotes, "cd ~/Desktop". Enter the command "ls" to list all of the files on the Raspberry Pi desktop to make sure that the PythonCode.py file is there. Lastly, enter the command "python PythonCode.py" to run the python code.

6.4 Subsystem 4 Data Logging

6.4.1 BOM

Table 6. Bill of Materials for Data Logging

Item	Link	Cost
Raspberry Pi 4	Raspberry Pi	$47.75

6.4.2 Equipment List

Latest Arduino IDE (software)

Provided Arduino Code

Latest Python 3 (software)

Provided Python code

Excel (software)

6.4.3 Instructions

Step 1.

If the Raspberry Pi, Arduino Uno, and MPU6050 are not already properly connected and turned on, then do that first. Make sure the Arduino is running the INO code that was given. Then, download the Python code from the provided github link and save the file onto the Raspberry Pi.

Step 2.

Open up any Python 3 software on the Raspberry Pi and run the Python code you downloaded and saved onto the Raspberry Pi. At this point, you should see an output in the serial monitor that looks something like Figure 12. At this point Python has now created and opened a file on your Raspberry Pi and is now writing (logging) the data shown in the serial monitor into that file.

```
Stable, -0.39,0.48
Stable, -0.47,0.63
Stable, -0.98,0.58
Stable, -0.73,0.40
Stable, -0.13,0.35
Stable, -0.59,0.52
Stable, -1.15,0.36
Stable, -0.61,0.45
Stable, -0.40,0.44
Stable, -0.36,0.40
```

Figure 12. Example of Output in Python software

Step 3.

Once you have run the code/flight for as long as you needed/wanted, at anypoint you will then stop the code using a keyboard interrupt (normally in Python the keyboard shortcut is: ctrl+c). This will then stop your code and close your file that Python was writing and logging your data in.

Step 4.

Next you will need to find your file. The file will be saved as the exact date and time that you started running the program (code). Refer to Figure 13 for an example of how the file should look. Once you have found this file you can open it up and it should show the same data you saw in the serial monitor. Refer to Figure 14 for how the opened .csv file should look.

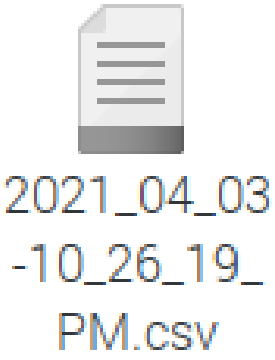

Figure 13. Example of .csv file closed

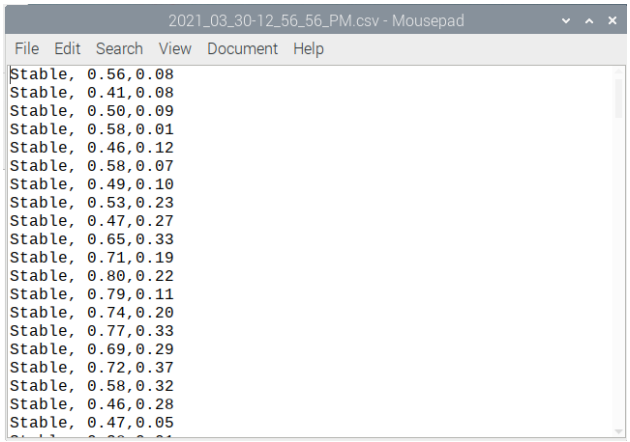

Figure 14. Example of .csv file opened

Step 5.

Once your file has been located you then can transfer the .csv file over to your pc.

Step 6.

If you do not have Excel installed on your pc, install Excel software at this point. opening the .csv file on your pc will open up an Excel spreadsheet with all your data. Refer to Figure 15 for how your spreadsheet should look.

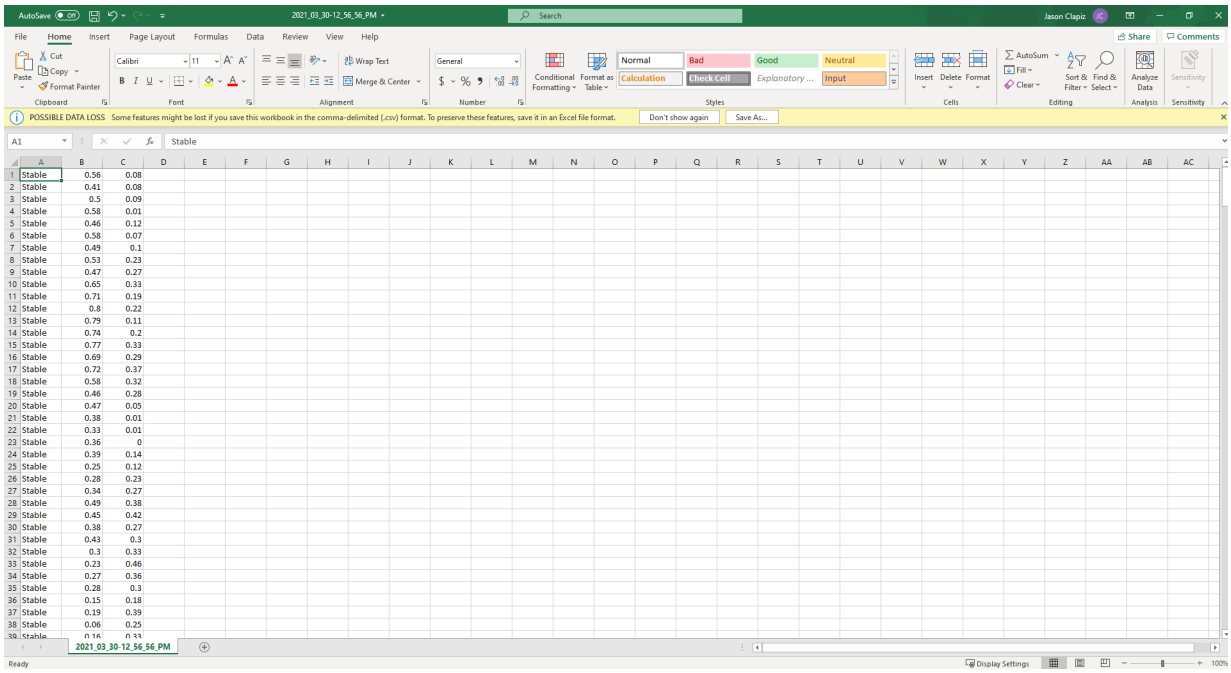

Figure 15. Example of Excel spreadsheet opened

Step 7.

Now you can play around with the excel spreadsheet and create a line chart to show the data. Refer to Figure 16 for an example of how your data can look once graphed.

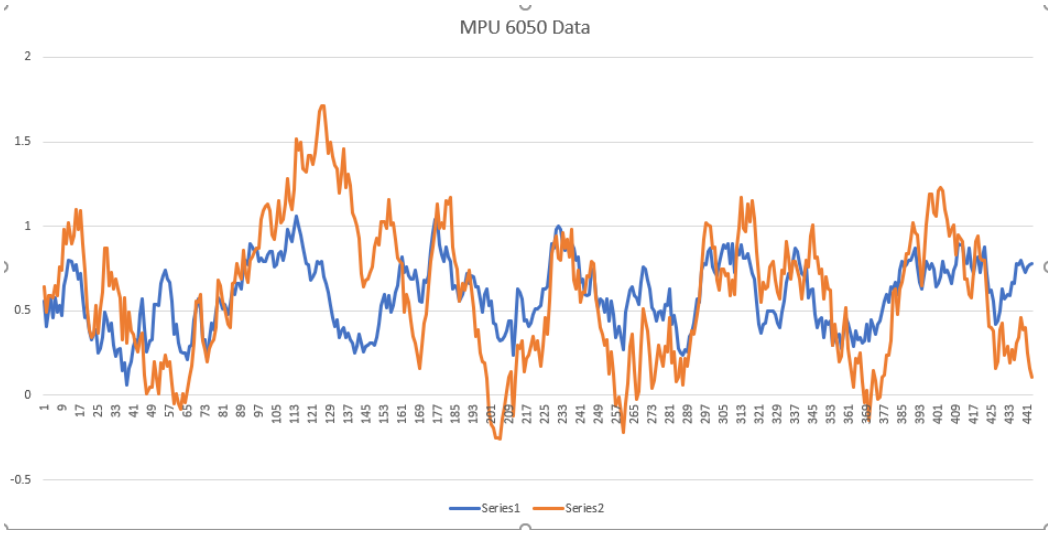

Figure 16. Example of MPU 6050 data in a graph

6.5 Optional Subsystem Device Housings

Not necessary to recreate a functional prototype

6.5.1 BOM

Table 7. Bill of Materials for Optional Subsystem Housings

Item	Link	Cost
3D Printing Filament	PLA Filament	$32.99/ Spool

6.5.2 Equipment List

3D Printer

3D printing software

6.5.3 Instructions

Step 1.

Recreate the housings in a 3D modelling software for each component based on the technical drawings provided below. Download the Python code that was provided from the

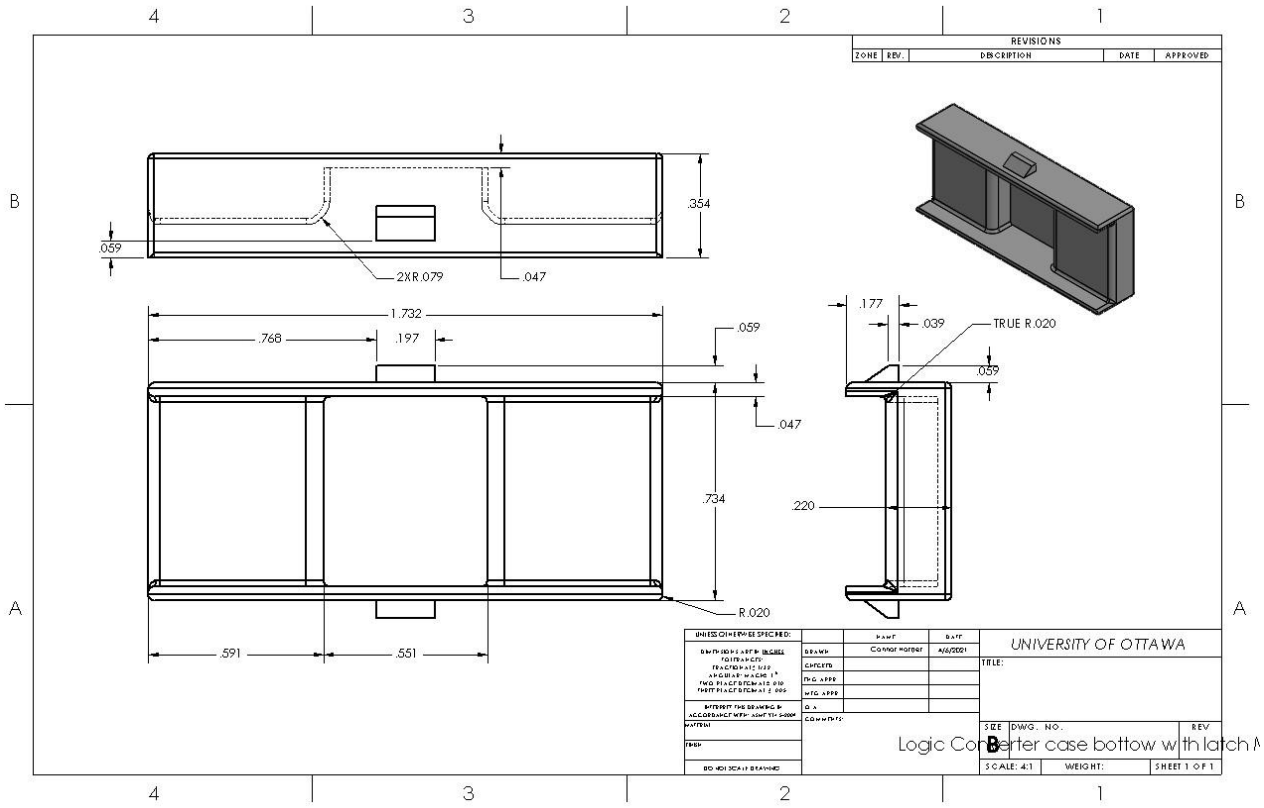

Fig 17. Bottom case housing for logic converter

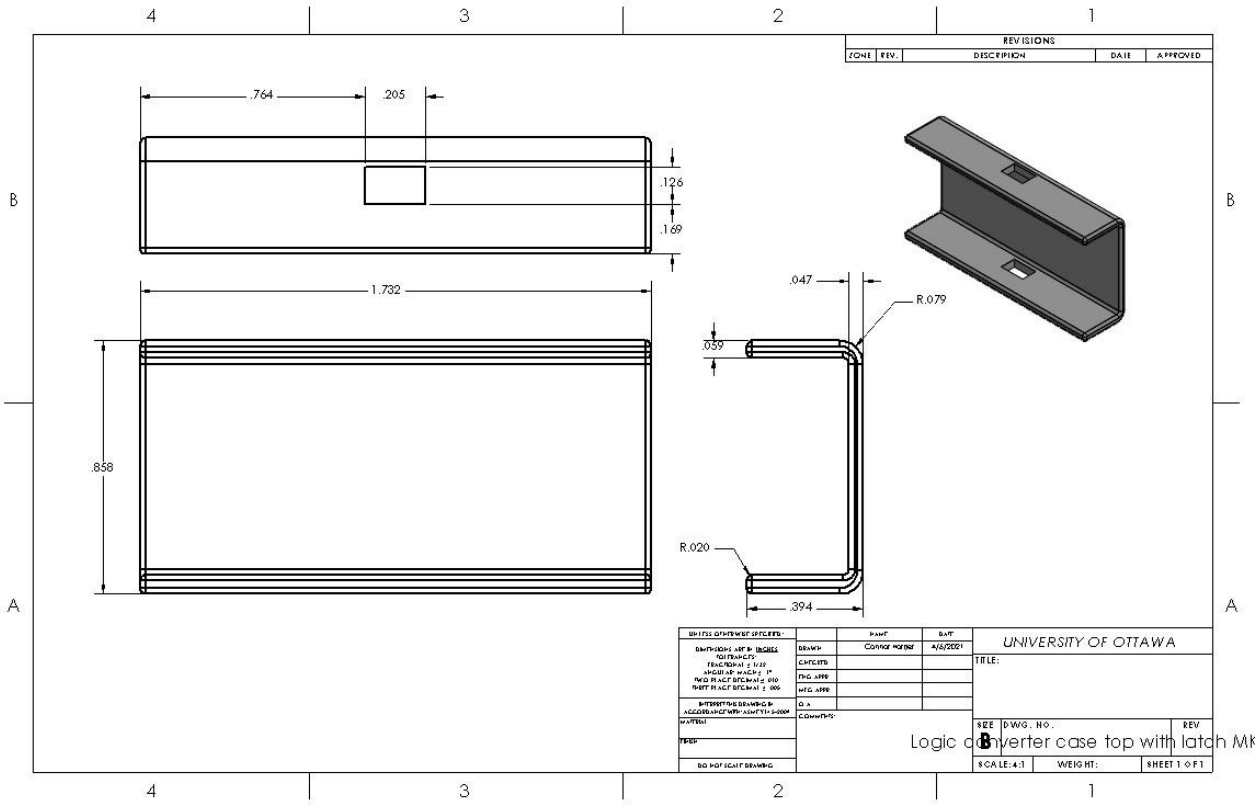

Fig 18. Top case housing for logic converter

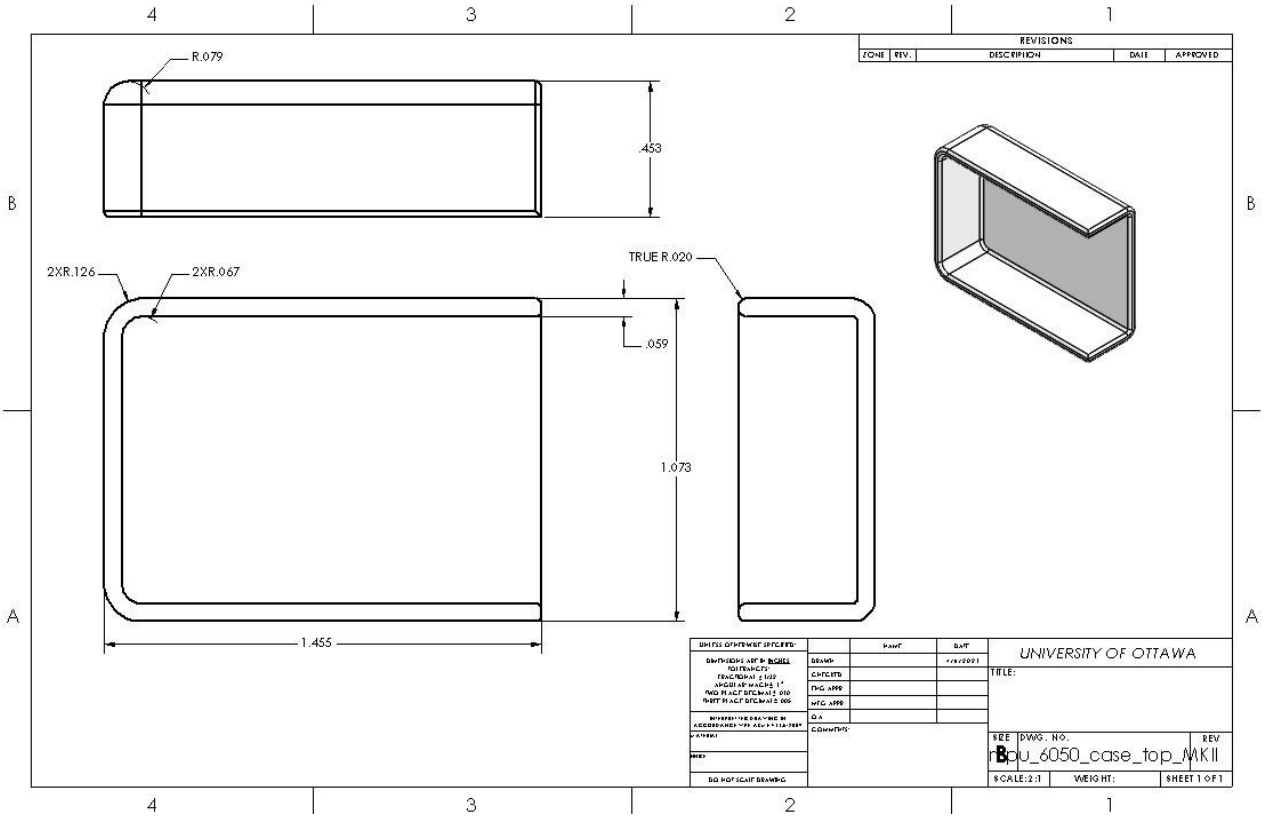

Fig 19. Top housing for MPU 6050

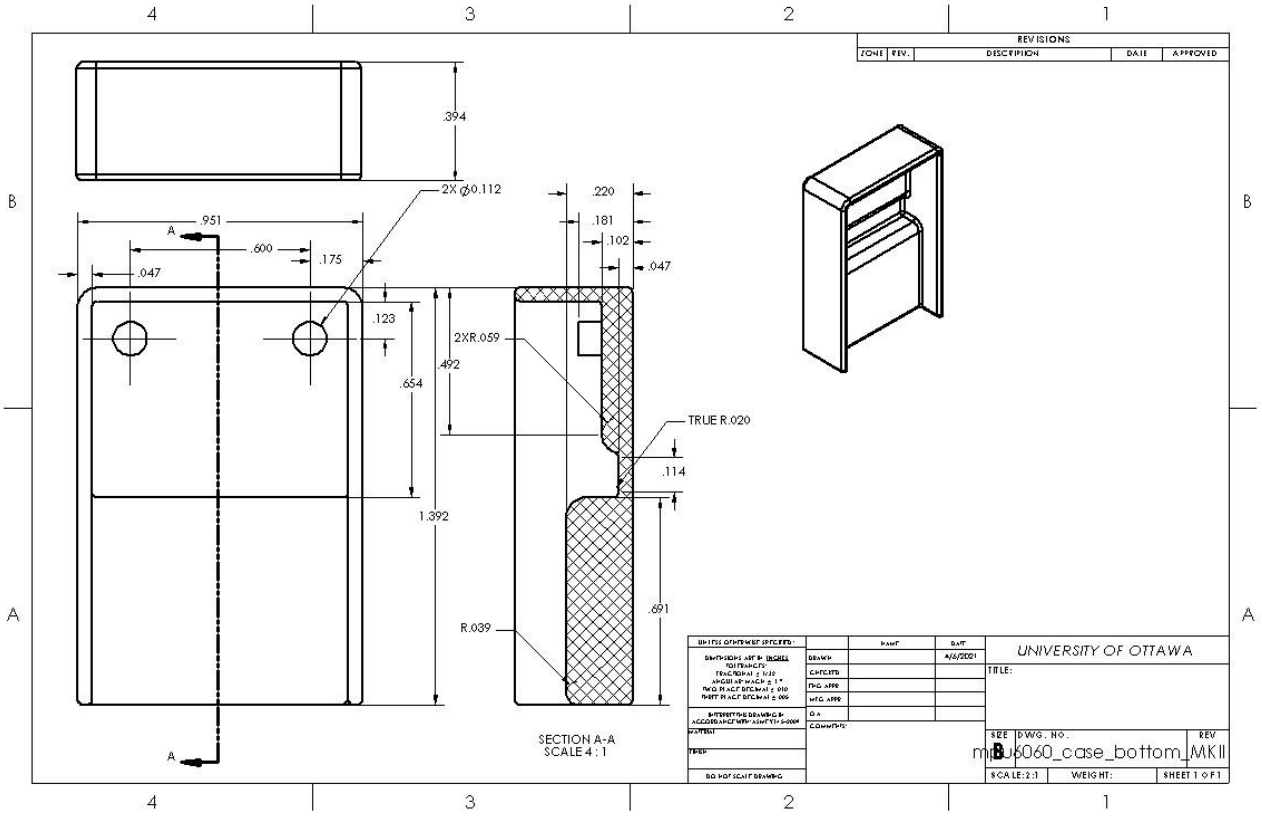

Fig 20. Bottom housing for MPU 6050

Step 2.

Export your files to .stl format.

Step 3.

Import your .stl files into your 3D printing software of choice and prepare your print.

Step 4.

Print the housings in PLA

6.6 Testing & Validation

Test 1. Proof of concept data collection

This test focused on subsystem 2, which consisted of validating the data collected from a sensor on the Arduino. With the sensors wired temporarily using pins and a breadboard, the Arduino displayed the data collected and changed with respect to the MPU 6050's movement. Therefore, this test validated our concept data collection subsystem.

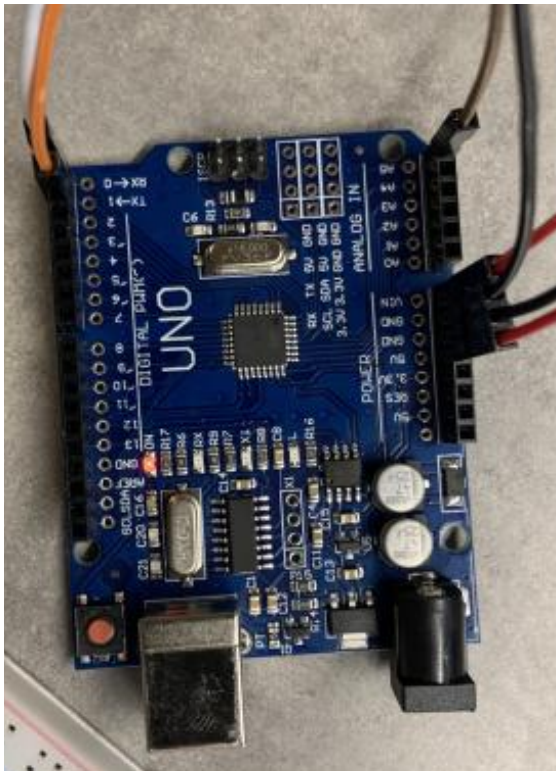

Figure 21. Arduino

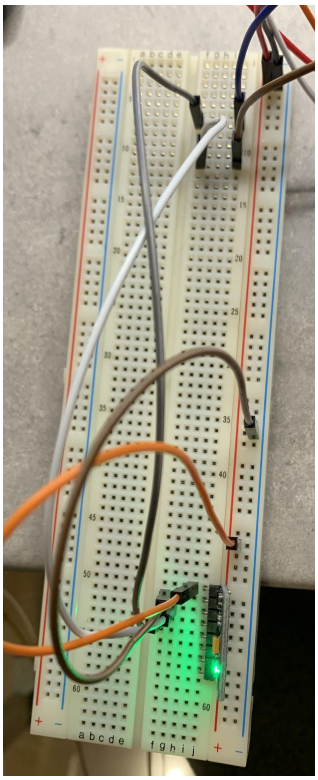

Figure 22. Breadboard

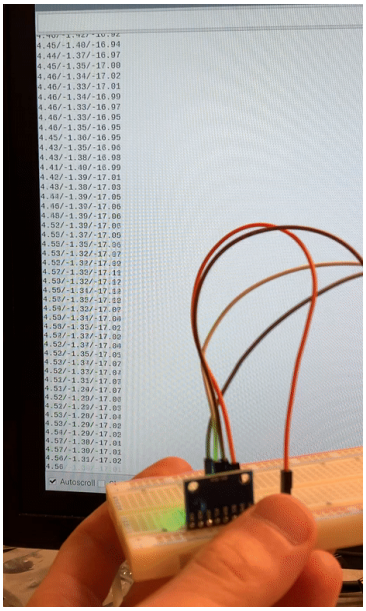

Figure 23. Data Transmission

Test 2. Proof of concept data transmission

This test focused on subsystem 3, which consisted of validating the data received on the Raspberry Pi. The MPU 6050 sensor was wired to the Arduino, which was then connected to the Raspberry Pi through a LC. Once the sensor was moved, providing an input, the test focused on reading the values on a serial monitor via the Raspberry Pi. Therefore, once the values were displayed the test was complete, validating our concept of data transmission.

Figure 24. Arduino

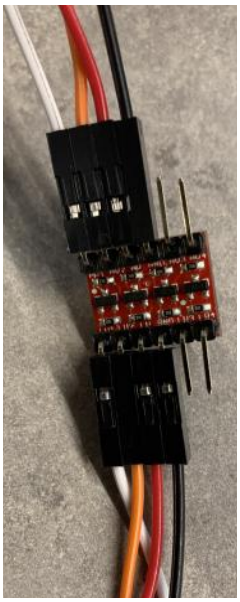

Figure 25. LC

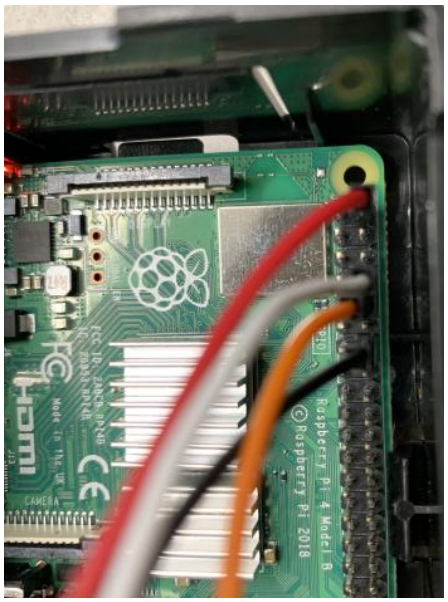

Figure 26. Raspberry Pi

Test 3. Interpreted Values

Referring to Figure 27, this test focused on defining the threshold ranges corresponding to stable, moderate, or violent shaking. Using qualitative measurements to represent the severity of shaking, those values were determined as the thresholds to updating the string response.

```
Stable      Moderate Shaking      Violent Shaking
```

Figure 27. Interpreted values

Test 4. Saving Flight Data

This test focused on saving the flight data as a .csv file which is stored on a SD card on the Raspberry Pi. The file will be stored under the active user when the software was started. The format of the .csv file will follow year, month, day, time of when the software was started. Refer to Figure 13 and Figure 14 for visuals of how the file should look when closed and opened.

7 Conclusions and Recommendations for Future Work

Summary of Lessons:

The main lessons that this team learned from this project were Communication, Team coordination, Time management, Prioritization, and Contingency Plans. The biggest, if not the biggest reason, that the project was successful was due to the constant, effective, and immediate communication among each team member. Since everyone on the team was always communicating with each other, they were able to do their own parts of the project as the project continued to progress without many setbacks. Since everyone's overall communication was always on point, the team's coordination was also well executed as each team member discussed and agreed on the realistic goal(s) that were set for those dates and times while everyone's personal life schedule was taken into consideration. This result lead to consistent and proper use of time usage during the team meetings, and the time spent on progressing the project was properly managed throughout each meeting. As the project progressed toward the first prototype, the team encountered a minor setback where a LED was used as a replacement for the MPU6050 sensor to check that data can be transmitted from LED to Arduino to Raspberry Pi, all due to the fact that the MPU6050 sensor was the only component that couldn't arrive on time. The LED replacement was a big prioritization as the team needed a physical working prototype to show the client, and the team decided to proceed with this working yet delayed working prototype for progressing further into the project. Lastly, the team encountered another set back where Deliverable C was not correctly done, so the team had to redo it as Deliverable D requires Deliverable C to be correctly finished. No one expected anything like this to happen, but everyone put extra effort into completing Deliverable C and D as they were the most crucial deliverables for the future prototyping deliverables.

More Time:

If more time was allotted to the development of this prototype, considerations would be given to creating a notification that would be sent to the client who ordered the food in the event of the violent shaking occuring during the flight. Research into wireless communication of data would have been interesting to work around. The potential to learn about these wireless options would have proven to be very beneficial but is outside the scope of the time required to develop this product.

8 Bibliography

.INO File Extension. INO File Extension - What is an .ino file and how do I open it? (2021, March 31). https://fileinfo.com/extension/ino.

Bi-Directional Logic Level Converter Hookup Guide. (n.d.). https://learn.sparkfun.com/tutorials/bi-directional-logic-level-converter-hookup-guide/all.

Dejan. (2021, February 5). *Arduino and MPU6050 Accelerometer and Gyroscope Tutorial*. HowToMechatronics. https://howtomechatronics.com/tutorials/arduino/arduino-and-mpu6050-accelerometer-and-gyroscope-tutorial/.

Fitzpatrick, M. (2021, April 9). *Plotting in PyQt5 - Using PyQtGraph to create interactive plots in your apps*. Learn PyQt. https://www.learnpyqt.com/tutorials/plotting-pyqtgraph/.

GPIO. GPIO - Raspberry Pi Documentation. (n.d.). https://www.raspberrypi.org/documentation/usage/gpio/#:~:text=A%20powerful%20feature%20of%20the,Zero%20and%20Pi%20Zero%20W).

Hrisko, J. (2019, November 25). *MPU6050 Arduino High-Frequency Accelerometer and Gyroscope Data Saver*. Maker Portal. https://makersportal.com/blog/2019/8/17/arduino-mpu6050-high-frequency-accelerometer-and-gyroscope-data-saver.

Monk, S. (n.d.). *Adafruit's Raspberry Pi Lesson 5. Using a Console Cable*. Adafruit Learning System. https://learn.adafruit.com/adafruits-raspberry-pi-lesson-5-using-a-console-cable/enabling-serial-console.

Oscar. (2018, March 23). *Raspberry Pi and Arduino Connected Over Serial GPIO*. Oscar Liang. https://oscarliang.com/raspberry-pi-and-arduino-connected-serial-gpio/.

Setup Raspberry Pi Hardware Permissions - The Robotics Back. End. (2020, June 22). https://roboticsbackend.com/raspberry-pi-hardware-permissions/.

Watkins, J. (2019, August 12). *Configuring The GPIO Serial Port On Raspbian Jessie and Stretch Including Pi 3 and 4*. Spell Foundry. https://spellfoundry.com/2016/05/29/configuring-gpio-serial-port-raspbian-jessie-including-pi-3-4/.

What is a .CSV file and what does it mean for my ecommerce business? BigCommerce. (2021, April 9). https://www.bigcommerce.ca/ecommerce-answers/what-csv-file-and-what-does-it-mean-my-ecommerce-business/#:~:text=A%20CSV%20is%20a%20comma,Microsoft%20Excel%20or%20Google%20Spreadsheets.

What is an IDE? (n.d.). https://www.redhat.com/en/topics/middleware/what-is-ide#:~:text=An%20integrated%20development%20environment%20(IDE,graphical%20user%20interface%20(GUI).

What is UART Protocol? UART Communication Explained. Arrow.com. (2020, March 6). https://www.arrow.com/en/research-and-events/articles/what-is-uart-protocol-uart-communication-explained#:~:text=Universal%20asynchronous%20receiver%2Dtransmitter%20(UART,ICs)%20or%20as%20individual%20components.

Wikimedia Foundation. (2021, March 13). *STL (file format)*. Wikipedia. https://en.wikipedia.org/wiki/STL_(file_format)#:~:text=1987,and%20%22Standard%20Tessellation%20Language%22.

APPENDICES

9 APPENDIX I: Design Files

The “deliverable” documents included below in table 8 progressively build on top of each other, in alphabetical order from B-J. The deliverables have individual focuses pertaining to the overall design process.

Table 8. Referenced Documents

Document Name	Document Location and/or URL	Issuance Date
Makerepo	https://makerepo.com/ConnorHarper/840.gng1103d6diamond-hands	Mar 15, 2021
Deliverable B	https://docs.google.com/document/d/1Lj7XM3FdZXyAfNJpMevaZX0mDNR_I-xjbY-vGX9xS9M/edit?usp=sharing	Jan 31, 2021
Deliverable C	https://docs.google.com/document/d/1xsBNRWlv6T4OBLY8CV6IdEB3hU24dsCuVeMGvUIkZ1Y/edit?usp=sharing	Feb 7, 2021
Deliverable D	https://docs.google.com/document/d/1L_edRZlOOTdwT55SU-SGZNinIeab9xJqyKf-JqrywcE/edit?usp=sharing	Feb 21, 2021
Deliverable E	https://docs.google.com/document/d/126upKvCQWXuLUPKrPNGL03U4kpOyeie0sdA9aZ0Tc54/edit?usp=sharing	Feb 28, 2021

Deliverable F	https://docs.google.com/document/d/1CeyUVcyK8yQ_38LjOjY327novW3uuQ83fEJQeiRNOvw/edit?usp=sharing	March 7, 2021
Deliverable G	https://docs.google.com/document/d/1xXMvCvd4nFpu7Tpr4sGph0JmIRzrTiAulJ0wWISLHpY/edit?usp=sharing	March 14, 2021
Deliverable H	https://docs.google.com/document/d/1Cw5TDMONGZS05tNay_wWRREleB-SSgeiLc1hK1dFzQg/edit?usp=sharing	March 28, 2021
Deliverable I	https://docs.google.com/presentation/d/1SPwVEv5JGYUbE3Bb4pgeicYEA63J3LaK7qU15o9-zp8/edit?usp=sharing	March 20, 2021
Deliverable J	https://docs.google.com/presentation/d/131XFc69X23ANlsjxa6jr0KygCYO51e59hESTF3DuhI0/edit?usp=sharing	March 22, 2021

10 APPENDIX II: Other Appendices

Github Link to Access and Download the Files

The following Arduino code is to be ran on the Arduino IDE (from the ArduinoCode.ino file):

```
/*
   MPU 6050 data collection and transmission for JAMZ drone delivery
   by Connor Harper, Jason Clapiz, Karsten Lowe, Leo Tan
   based on software originally written by Dejan https://howtomechatronics.com
*/
#include <Wire.h>
const int MPU = 0x68; // MPU6050 I2C address
float AccX, AccY, AccZ;//accelerometer reading
float GyroX, GyroY, GyroZ;//gyroscope reading
float accAngleX, accAngleY, gyroAngleX, gyroAngleY;//Accelerometer and gyroscope angle
float roll, pitch, yaw;//drone's position data variables
float AccErrorX, AccErrorY, GyroErrorX, GyroErrorY, GyroErrorZ;//calculated error during
MPU6050 self calibration
float elapsedTime, currentTime, previousTime;//tracks time based on when the code was
initialized
int c = 0;//count calibration readings

int moderateValue = 690, violentValue = 820; //threshold range value
void setup() {
  Serial.begin(19200);//baud rate for the serial monitor
  Wire.begin(); // Initialize communication
  Wire.beginTransmission(MPU); // Start communication with MPU6050 // MPU=0x68
  Wire.write(0x6B); // Talk to the register 6B
  Wire.write(0x00); // Make reset - place a 0 into the 6B register
  Wire.endTransmission(true); //end the transmission
  calculate_IMU_error();//runs the error calculation function before the main code executes
}
void loop() {
  // === Read acceleromter data === //
```

```
  Wire.beginTransmission(MPU);//starts communication with the MPU 6050
  Wire.write(0x3B); // Start with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);//prevents the communication from ending
  Wire.requestFrom(MPU, 6, true); // Read 6 registers total, each axis value is stored in 2
registers
  //For a range of +-2g, we need to divide the raw values by 16384, according to the datasheet
  AccX = (Wire.read() << 8 | Wire.read()) / 8192.0; // X-axis value
  AccY = (Wire.read() << 8 | Wire.read()) / 8192.0; // Y-axis value
  AccZ = (Wire.read() << 8 | Wire.read()) / 8192.0; // Z-axis value
  // Calculating Roll and Pitch from the accelerometer data
  accAngleX = (atan(AccY / sqrt(pow(AccX, 2) + pow(AccZ, 2))) * 180 / PI) - AccErrorX;
//calculates the acceleration in the X axis while taking into account the error found during
self calibration
  accAngleY = (atan(-1 * AccX / sqrt(pow(AccY, 2) + pow(AccZ, 2))) * 180 / PI) -
AccErrorY;//calculates the acceleration in the Y axis while taking into account the error found
during self calibration
  // === Read gyroscope data === //
  previousTime = currentTime; // Previous time is stored before the actual time read
  currentTime = millis(); // Current time actual time read
  elapsedTime = (currentTime - previousTime) / 1000; // Divide by 1000 to get seconds
  Wire.beginTransmission(MPU);// Start communication with MPU6050 // MPU=0x68
  Wire.write(0x43); // Gyro data first register address 0x43
  Wire.endTransmission(false);//prevents the communication from ending
  Wire.requestFrom(MPU, 6, true); // Read 4 registers total, each axis value is stored in 2
registers
  GyroX = (Wire.read() << 8 | Wire.read()) / 131.0; // For a 250deg/s range we have to divide
first the raw value by 131.0, according to the datasheet
  GyroY = (Wire.read() << 8 | Wire.read()) / 131.0;
  GyroZ = (Wire.read() << 8 | Wire.read()) / 131.0;
  // Correct the outputs with the calculated error values, the error will self correct based on
the calibration when the system was initialized
  GyroX = GyroX - GyroErrorX;
  GyroY = GyroY - GyroErrorY;
  GyroZ = GyroZ - GyroErrorZ;
  // Modifies the stored gyroscope angles based on the previous recorded angles
```

```
  gyroAngleX = gyroAngleX + GyroX;
  gyroAngleY = gyroAngleY + GyroY;
  yaw = yaw + GyroZ;
  // Complementary filter - combine acceleromter and gyro angle values
  gyroAngleX = 0.96 * gyroAngleX + 0.04 * accAngleX;
  gyroAngleY = 0.96 * gyroAngleY + 0.04 * accAngleY;
  //Renames effective variables for ease of use during angle filtering
  roll = gyroAngleX;
  pitch = gyroAngleY;

  //checks read values against set values that constitute violent or moderate shaking
  //if one of the conditions is met, the code skips over the rest in an effort to optimise run
time.
  if (roll > violentValue || roll < -violentValue || pitch > violentValue || pitch <
-violentValue) {
    Serial.print("Violent_Shaking, ");
  }
  else if (roll > moderateValue && roll < violentValue){
    Serial.print("Moderate_Shaking, ");
  }
  else if(roll < -moderateValue && roll > -violentValue){
      Serial.print("Moderate_Shaking, ");
  }
  else if(pitch > moderateValue && pitch < violentValue){
      Serial.print("Moderate_Shaking, ");
  }
  else if(pitch < -moderateValue && pitch > -violentValue) {
      Serial.print("Moderate_Shaking, ");
  }
  else {
    Serial.print("Stable, ");
  }
  printValues(roll, pitch, yaw);
}
void calculate_IMU_error() {
```

```
  // This funtion is called in the setup section to calculate the accelerometer and gyro data
error
  while (c < 200) {// Read Accelerometer values 200 times
    Wire.beginTransmission(MPU);
    Wire.write(0x3B);
    Wire.endTransmission(false);
    Wire.requestFrom(MPU, 6, true);
    AccX = (Wire.read() << 8 | Wire.read()) / 8192.0;//divide by 8192 to get the proper axis
value based on data sheet
    AccY = (Wire.read() << 8 | Wire.read()) / 8192.0;
    AccZ = (Wire.read() << 8 | Wire.read()) / 8192.0;
    // Sum all readings using trig to calculate the accelerational error
    AccErrorX = AccErrorX + ((atan((AccY) / sqrt(pow((AccX), 2) + pow((AccZ), 2))) * 180 /
PI));
    AccErrorY = AccErrorY + ((atan(-1 * (AccX) / sqrt(pow((AccY), 2) + pow((AccZ), 2))) * 180 /
PI));
    c++;
  }
  //Divide the sum by 200 to get the error value
  AccErrorX = AccErrorX / 200;
  AccErrorY = AccErrorY / 200;
  c = 0;
  // Read gyro values 200 times
  while (c < 200) {
    Wire.beginTransmission(MPU);
    Wire.write(0x43);
    Wire.endTransmission(false);
    Wire.requestFrom(MPU, 6, true);
    GyroX = Wire.read() << 8 | Wire.read();
    GyroY = Wire.read() << 8 | Wire.read();
    GyroZ = Wire.read() << 8 | Wire.read();
    // Sum all readings
    GyroErrorX = GyroErrorX + (GyroX / 131.0);
    GyroErrorY = GyroErrorY + (GyroY / 131.0);
    GyroErrorZ = GyroErrorZ + (GyroZ / 131.0);
```

```
    c++;
  }
  //Divide the sum by 200 to get the error value
  GyroErrorX = GyroErrorX / 200;
  GyroErrorY = GyroErrorY / 200;
  GyroErrorZ = GyroErrorZ / 200;
}

//print the calculated values into the serial monitor
void printValues(float roll, float pitch, float yaw) {
  Serial.print(roll);
  Serial.print(",");
  Serial.println(pitch);
  //Serial.print(",");
 // Serial.println(yaw);
}
```

The following Python code is to be ran on the Python IDE (from the PythonCode.py file):

```
import serial
from datetime import datetime
if __name__ == '__main__':
    flag = 0
    current_time = datetime.now().strftime("%Y_%m_%d-%I_%M_%S_%p")
    file = open ( str(current_time)+".csv", 'w' )
    ser = serial.Serial('/dev/ttyS0', 19200, timeout=1)
    ser.flush()
    while True:
        if ser.in_waiting > 0:
            line = ser.readline().decode('utf-8').rstrip()
            print(line)
            file.write (str(line))
    file.close()
```

Download the printable files for the MPU6050 and the Logic Converter.