Final report

Miguel Velasco Espinosa

Texas Tech

December 2022

Abstract

This paper will show the use of hardware and software to launch a model rocket and collect data such as pressure, humidity, temperature, altitude, and launch force. The hardware required is the raspberry pi which will be used to host almost all the software used which is maria database and python. During the 8-week period a successful rocket launch without the raspberry pi was completed and the software side is near completion.

Table of contents

List of Figures…………………………………………………………………………………….3

List of Tables……………………………………………………………………………………...4

Introduction………………………………………………………………………………………..5

Project Opportunities……………………………………………………………………………...5

Hardware…………………………………………………………………………………………..6

Software………………………………………………………………………………………….18

Results……………………………………………………………………………………………23

Conclusion……………………………………………………………………………………….28

Ethics & Safety…………………………………………………………………………………..29

Acknowledgements………………………………………………………………………………30

References………………………………………………………………………………………..31

Appendix…………………………………………………………………………………………32

List of figures

Fig. 1 Raspberry Pi Zero W……………………………………………………………………….6

Fig. 2 Raspberry Pi zero W GPIO Pin Layout…………………………………………………….7

Fig. 3 Pololu Buck Convertor……………………………………………………………………..8

Fig. 4 Buck Converter Connections on Raspberry Pi 0 W………………………………………..8

Fig. 5 K380 keyboard & M350 Mouse combo……………………………………………………9

Fig. 6 Dell S2716DG…………………………………………………………………………….10

Fig. 7 BME280…………………………………………………………………………………..11

Fig. 8 H3LIS331DL……………………………………………………………………………...12

Fig. 9 H3LIS331DL 3-Axis Detection…………………………………………………………...12

Fig. 10 Raspberry Pi Zero W sensor layout……………………………………………………...13

Fig. 11 Payload Hardware………..………………………………………………………………14

Fig. 12 Rocket 1………………………………………………………………………………….14

Fig. 13 Rocket 2…………………………...……………………………………………………..15

Fig. 14 Black Max Rocket Motors……………………………………………………………….15

Fig. 15 Estes Low Power Launch System……………………………………………………….16

Fig. 16 Second Launch Pad……………………………………………………………………...17

Fig. 17 Code Flowchart…………………………………………………………………………..18

Fig. 18 H3LIS331DL Code snippet……………………………………………………………...20

Fig. 19 Python code test………………………………………………………………………….22

Fig. 20 Altitude Graph..………………………………………………………………………….23

Fig. 21 Temperature Graph………………………………………………………………………24

Fig. 22 Humidity Graph………………………………………………………………………….25

Fig. 23 Pressure Graph…………………………………………………………………………...26

Fig. 24 Launch Force Graph……………………………………………………………………..27

List of Tables

Table I Database Columns……………………………………………………………………….21

1. Introduction

The project’s purpose is to use a model rocket that will be able to gather atmospheric data using a raspberry pi and successfully use maria database to keep that gathered data stored. To collect the atmospheric data a BME280 will be used. The BME280 can collect humidity, pressure, altitude, and temperature. The project also requires calculation of launch force so a H3LIS33IDL accelerometer will be used as it can calculate the acceleration which then can be used to calculate the launch force. The main programing language used is python 3 as it is the required language and it will be used to program the raspberry pi, the BME280, and the H#LIS33IDL. To set up the database SQL will be used to add information to the maria database. An extra hardware part not required by the project is, as of the writing of this paper, the Optimus 2.0 GPS which is used for locating the rocket after launch.

* 1. Project Opportunities

One of the main areas this project will be extremely useful to is atmospheric data gathering. Weather balloons are the one of the main sources to gather high-altitude data however the take up to 2 hours to gather the data and might drift hundreds of miles away. With the rocket it has a much shorter area to which it can travel to and will be easier to retrieve and easier to deploy. The other area which will benefit from this is space exploration specifically gathering data from other planets in the solar system. The main area this rocket project will hit would be cheap & efficient ways to gather atmospheric data since it can be launched from pretty much everywhere it would be able to be deployed from a bigger rocket and send back data collected while on the atmosphere.

1. Project Description

Build a model rocket that will collect launch data from a series of launches, such as temperature, humidity, barometric pressure, altitude, & launch force. Data will be stored in a Raspberry Pi Maria database that is housed in the rocket. All rocket launches will be performed at Lt. Col. George Andrew Davis Jr. Field, 15th Street and Inlet Ave (FM 179). Students should be able to demonstrate basic SQL (Structured Query Language) gymnastics.

1. Hardware

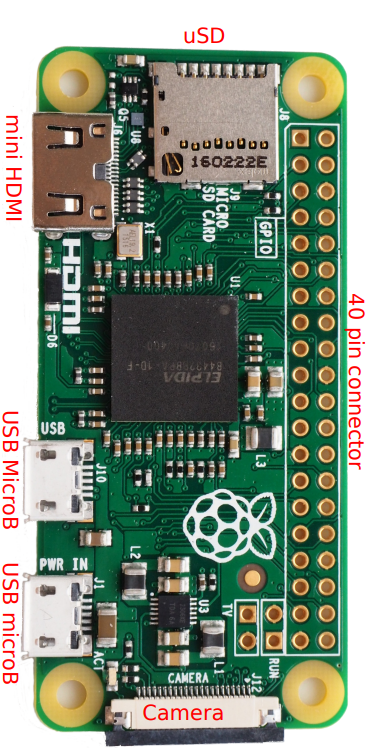
 The project requires a raspberry pi but never specified which kind. After researching different varieties of raspberry pi, it was concluded that the best suited was the Raspberry Pi Zero W. This Raspberry Pi is small enough to be able to fit into the rocket purchased for this project ,which will be discussed later in the paper, and is able to host the maria database and run the python scripts to gather the information required.

Fig. 1 Raspberry Pi Zero W [Powell 1]

The Raspberry Pi Zero W ,which from this point forward will be referred to as RPZ, can be powered by a 5-volt source in the “PWR IN” USB microB as show in Fig. 1. The board is usually powered on by connecting the micro USB to a computer or wall outlet however for the purposes of this project a new way to power it needs to be researched. As of the writing of this paper the research has not been completed on powering up the raspberry pi while on the rocket.

To host the Raspberry Pi OS an SD is required which goes on the uSD spot in fig. 1. In this project a 32GB SD card was utilized to store the OS for the RPZ. When powering up the board the OS is loaded via the SD card.

Timeline

Description automatically generated with medium confidenceThe RPZ has a lot of GPIO pins which are important for the setup of the RPZ. This is where the accelerometer and the BME will be connected to. As well as possible other parts needed for the project.

Fig. 2 Raspberry Pi zero W GPIO Pin Layout[Powell 2]

Graphical user interface, application

Description automatically generated To be able to power up the board while on the rocket without the PWR IN USB (Fig 1.) there were many ideas as to how it would be possible, and it was decided a Pololu Buck Convertor was decided.

Fig. 3 Pololu Buck Converter

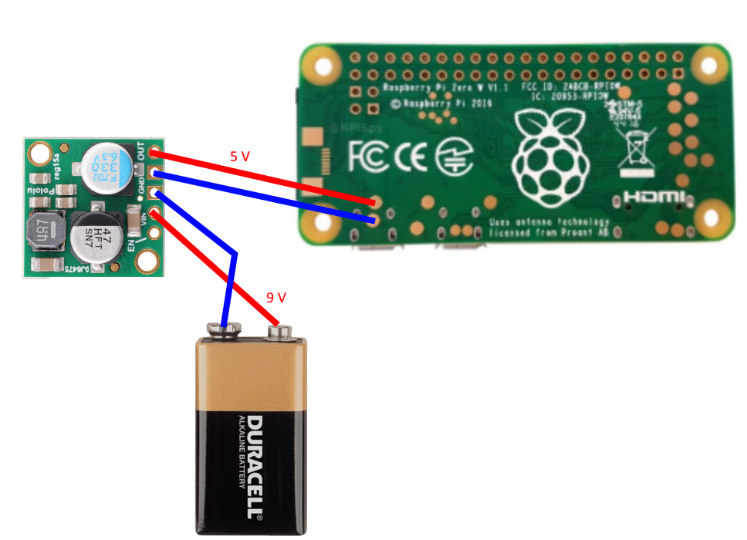
The buck convertor was needed since you can’t power up the RPZ with just a 9 volt battery otherwise it would burn the RPZ. The buck convertor outputs up to 2.5 amps while the RPZ only needs 1 – 2 amps. The buck convertor was also desired because of its small size since it is only 0.7 inches by 0.7 inches meaning putting it into the payload was not a big concern.

Fig. 4 Buck Convertor Connections on Raspberry Pi 0 W

The buck convertor takes the 9 volts and transforms it into 5 volts. The buck convertor is connected to GPIO ports and can give it power. An issue with this approach is that the RPZ would restart if the battery fell to 8.25 volts. This made it harder to keep only one battery for many flights so a battery change would have to happen every 2 to 3 flights.

To be able to interface with the RPZ a wireless keyboard and mouse was required the one used in this project is the K380 Multi-device keyboard & M350 Pebble Mouse. The reason this combo was selected is because they are wireless devices that only need one USB connector. A purchase of a USB to micro USB adaptor also needed to be purchased.

Fig. 5 K380 keyboard & M350 Mouse combo[Logitech 1]

Using the mouse and keyboard combo makes it possible to use the RPZ and be able to code it. A monitor was also required and for the purposes of this project and since the RPZ requires a HDMI cable a Dell S2716DG 27 inch monitor was used as it was already in the possession of the group and met all the requirements. The purchase of a mini HDMI to HDMI cable was needed as the RPZ has a mini HDMI port so a cable was needed to be able to see the RPZ with the monitor.

Fig. 6 Dell S2716DG [Dell 1]

After gathering all the necessary hardware for the RPZ the next step was to examine which components would be best to achieve the goals for the project. As stated before in this paper the BME280 was chosen to be the best choice since it gathered almost all the data needed for the project.

A picture containing text, electronics, blue

Description automatically generatedFig. 7 BME280 [Powell 3]

The BME280 can gather data about pressure, humidity, altitude, and temperature. The sensor in the BME280 requires 3 volts to be able to function, however a voltage regulator is in the BME280 that is connected to the VIN pin (Fig. 5) allows the voltage given to the BME280 to be in the range of 3 to 5 volts. Because of this the BME280 can be powered by the RPZ using the GPIO pins(Fig. 2). The RPZ supports SPI and I2C logic, for the purpose of this project I2C logic will be used. Since I2C will be used two pins in the RPZ will be used to send data these pins are the ”SCK” and “SDI”(Fig. 2).

The next component that compliments the BME280 is the H3LIS33DL, which will be referred to as H3L from this point forward, the H3L is an accelerometer that will collect the acceleration of the rocket and with that be able to calculate the launch force for the rocket.

A picture containing text, electronics

Description automatically generatedFig. 8 H3LIS331DL [SparkFun 1]

Diagram, engineering drawing

Description automatically generated The H3L (Fig. 6) uses a maximum of 3.6 volts so because of the requirements from both the BME280 and the H3L the RPZ only needs to give 3.3 volts to power both. The H3L sensor measures acceleration in a 3 dimensional plane. This means that the readings will be in the X, Y, and Z plane. The accelerometer code written for this project will be discussed later in the paper. The acceleration will be calculated in two ways in G’s and in .

Fig. 9 H3LIS331DL 3-Axis Detection[Powell 4]

As previously stated, the H3L detects the speed of 3 axis this data will then be transformed into G’s and then into . The average of the axis combined will then let the RPZ calculate the launch force after a certain amount of time has passed. As of the writing of this paper that implementation has yet to be completed.

A picture containing text, electronics, circuit

Description automatically generatedFig. 10 Raspberry Pi Zero W sensor layout.[Powell 5]

The BME280 and the H3L are set up with a I2C connection and are also daisy chained together. This helps the data be transmitted through only one bus instead of multiples buses. As well as using only one power pin which is the 3.3v. The connection has had no issues as of the writing of this paper.

A circuit board with wires

Description automatically generated with low confidenceFig. 11 Payload hardware

A picture containing text, stationary

Description automatically generated The payload hardware (Fig. 11) is compacted with tape to be able to fit the payload of the rocket. The tape also helps the connections stay put together and not disconnect. Many times, during the testing of this project wires would disconnect or break and after this set up the wires stop disconnecting and it was easier to retrieve the RPZ.

Fig. 12 Rocket 1 [Powell 6]

There where many rockets which could have completed the guidelines for the project however because of price and experience a cheap easy to construct rocket was chose which is the Apogee Superbird(Fig. 12). The Superbird costs around $22 USD it has a payload internal diameter of 1.35 inches and an internal payload length of 6 inches. Two of these rockets were purchased in case of fault or destruction of one of them the project wouldn’t be delayed waiting on a new rocket. This means that the payload is big enough to fit the RPZ as well as other components without issue. The Superbird has two parachutes, and it is meant to split the rocket into two parts with their own parachutes. The Superbird for this project was configured to keep the two sides attached and use only 1 parachute. The Superbird’s altitude depends a lot on the A picture containing text, writing implement, stationary, pen

Description automatically generatedtypes of motors used as well as the weight of the payload.

Fig. 13 Rocket 2

The two rockets weight 79 grams each without a payload and with the payload it weights around 150 grams.

Website

Description automatically generatedFig. 14 Black Max Rocket Motors[Apogee 1]

The Superbird can use 18 mm motors and for the purposes of this project the Black Max motors where used. They were selected for their price and their altitude as it would get a more distinct data set than other less powerful motors. The motors weight approximately 340 grams however because of their weight they take up of the Superbird’s maximum capacity. This adds more constraints to what can be in the payload of the rocket since too much weight might make it unpredictable and dangerous. The motors work by launching the rocket, then afterwards a small explosion occurs that releases the parachutes. Each pack contains 2 motors so 5 pack were purchased to test many rocket launches.

A picture containing text, indoor

Description automatically generatedFig. 15 Estes Low Power Launch System[Velasco 1]

To launch the rocket a power launch system was needed. The Apogee website ,where the Superbird was purchased, recommended the Estes Low Power Launch system(Fig. 13). The rocket is launched when the controller has the key inserted and then the button pressed. There are two metal prongs that send electricity through the motor making it spark. The metal rod helps the rocket stand and essentially helps to aim the rocket. The controller has a big enough cable where it is at a safe distance to launch the rocket.

Fig. 16 Second Launch pad

In the later parts of the project the first launch system was lost so a new one was needed. The second launch pad works the same as the first launch pad. The only difference is the size since the second launch pad is bigger than the first. The second launch pad also had a bigger base which helps with keeping it steady and have a better launch.

1. Software

Diagram

Description automatically generatedThe software’s main language is Python 3 and it is coded into the RPZ. At the writing of this paper the main objective is to have a main python script that will start whenever the RPZ turns on. The main script will then add the data collected during the launch to the maria database inside the RPZ.

Fig. 17 Code Flowchart[Powell 7]

The flowchart(Fig. 14) shows how the code will operate. First it starts by connecting to the database. There are libraries in python that connect the script to databases but for the purposes of this project the library pymysql will be used. After the connection a cursor is created. The cursor allows the addition of rows or change to the database by python code so controlling the database through the RPZ is possible. Afterwards a continuous loop goes on until the RPZ is shut down. The script first collects the data received from the BME280 then it will collect the H3L data and calculate the launch force. The script will then insert the data into the database and its respective rows. It will then wait a couple milliseconds to repeat the whole process after the loop is started. One of the conflicts was for how long the RPZ should run since having it run always makes it collect useless data. The main counter to that point is that the time the rocket spends on the air will be short so always keeping the RPZ on during the rocket launch is not a big issue.

The main concern with the programming was that at the beginning of the project it was believed that the RPZ would need to directly connect to a server and the maria database would be run off a laptop on the ground. A lot of resources and time were wasted on this pursuit, since it was later confirmed after going over the description again it was never specified how to store the data apart from the use of python and maria database it was decided to simply run the database inside the RPZ.

Text

Description automatically generatedFig. 18 H3LIS331DL Code snippet[Velasco 2]

The code for the H3L was copied from online and it was made to detect 24G’s maximum, which would not work for the project since model rockets can reach 100th’s of G’s, so a fix was found online which makes it possible to go up to 400G’s of detectable acceleration. However, the code fix increases a lot of the sensitivity of the H3L meaning in testing the G’s recorded without it moving would be 3 – 5 G’s which should not happen when it is not moving. Because of this additional testing and code modification was required this made it so that the H3L will be reading real values and was able to fix the sensitivity issue.

The BME280 code had a similar issue where it would not detect the right altitude and would detect the wrong pressure. A fix was found online where the pressure would be calculated at the start of the script and because of how the BME280 works it uses that pressure to calculate its altitude, so it fixed the calibration.

Writing the SQL also had its issues with specifically the formatting of the database and how to add items to it. Since the project requires maria database the maria database package was downloaded instead of the mySQL package. This caused some delay because although there is not much difference between using mySQL and using it with maria database the python library pymysql in addition to the small changes to the SQL code made it hard to find proper resources online to help with the project. However, after figuring out how the cursor works and how the python code talks to the database creating the table and adding data was quickly managed.

When writing test code for the BME280 it was also discovered that the RPZ was very slow with both gathering data and putting that data into the database. More proper adjustments need to be made to either the code or hardware to decrease the time the RPZ takes to perform the operations.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Date | Time | Temperature | Humidity | Pressure | Altitude | Launch Force |

Table I Database Columns[Velasco 1]

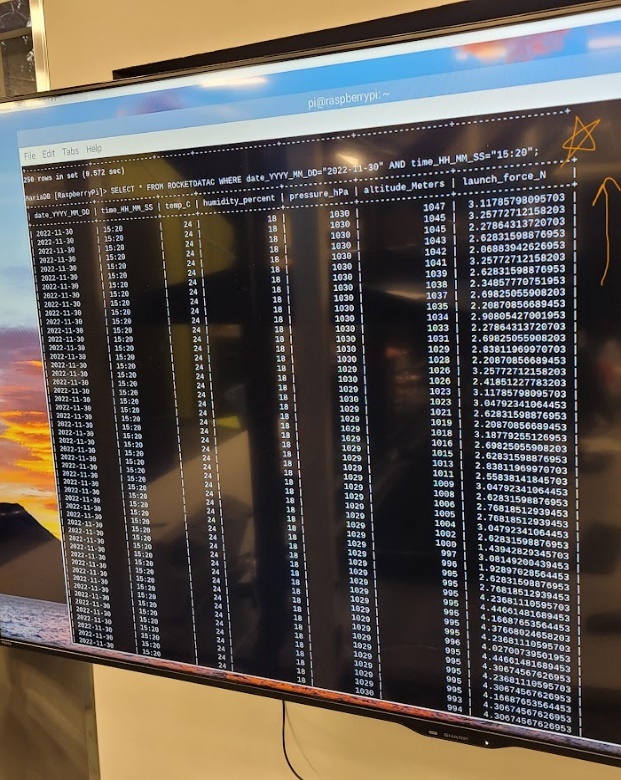
Table I is the rows of data that the table BME280 will have. The table stores all the data recorded whenever the rocket launch occurs meaning it will keep adding data to the table starting with the date and the time. The way the table is set up will make it easy to see the data of a specific date since all needed to look up the date and time is a simple SQL command. 

Fig. 19 Python code output[Powell 8]

Everything except date, time, and launch force is in integers. It was decided this was the best approach since it would be easier to check when the rocket went above a certain altitude making it easier to search when the rocket was launched. The range of the launch force also changed a lot through the project, and it will be shown later this paper.

1. Results

Chart, line chart

Description automatically generatedMany launches were conducted and that data was transferred to graphs to understand how the rocket behaves and the data it gathered. There were a total of 7 launches where data was recorded however only the last 3 have a correct launch force since the launch force before those launches was being calculated wrong.

Fig. 20 Altitude graph

All the 7 flights altitude was recorder in feet (Fig. 20). The longest flight was launch 4 which lasted about 30 seconds. The max altitude for launch 4 was around 430 feet. All the launches reached at least 200 feet and they stay in between the range 200 to 500 feet. During the gathering of data rocket 1 (FIG SOMETHING DON’T FORGET) there was a malfunction and rocket 1 was destroyed due to what is assumed a faulty engine. Rocket 1 was launched 3 times and follow launch 1, 2, and 3. After launch 3 Rocket 2 was used and there was some variety between rocket 1 and 2 since when looking at the graph you can see Rocket 1 launches 2 and 3 didn’t go higher than 300 feet. Rocket 1 may have had some construction errors and caused it to not go as high as possible unlike Rocket 2. If this project must be replicated a different rocket might be better suited for such a project.

Chart

Description automatically generated The highest altitude reach was 459 for launch 1. The day when this launch was made was very different from the other launches because of the weather the rocket managed to get up to 459 feet. Launch 4 had the longest time because the wind speed was 22 mph this helped the rocket stay up in the air for longer time compared to the other launches which had a wind speed below 20 mph. Other factors that affected the rocket launches where faulty engines which for this project there where 16 engines and 2 of them malfunctioned, the launch pad also affected the rocket as the second launchpad had a bigger area and didn’t tilt as much as the first.

Fig. 21 Temperature graph

Chart, line chart

Description automatically generated The temperature readings line up with what was expected that being a change of around 1 Celsius per 500 feet. That’s why the graph sees very little change.

Fig. 22 Humidity graph

The humidity graph shows the humidity of each launch and because most launches were done in different days they don’t line up that well. However, when checking the weather the percent was pretty accurate in between 2% of the actual humidity.

Chart, line chart

Description automatically generatedFig. 23 Pressure Graph

The pressure as shown in most of the launches has a lot of spikes this is due to the launching of the rocket the pressure inside the payload would drastically increase however after the rocket was fire pressure returned to normal and the reading correlate to what the pressure read on the weather channel said.

Chart, histogram

Description automatically generatedFig. 24 Launch Force

As previously stated the launch force was only recorded for the last 3 launches since it wasn’t properly calculated in the code. The graph shows that there where two spikes in most of the launches the first spike correlates with the launch of the rocket while the second is when the parachute opens and slows the rocket down. The launch force lines up with the engine data which says that when the parachute would deploy a max of 22 newtons would be expected. In Fig.21 the data correlates to what the engines described.

The results from the launches where a success in gathering atmospheric data and being able to use the database to look at how the different measurements compared with each other. The RPZ after the 7th launch stopped recording data and an extra launch was lost. The RPZ did not start the code on when it booted up like the rest. However, this was not that big of a loss since there where no more engines and sufficient data was gathered to form a proper paper.

1. Conclusion

The purpose of the project is to collect atmospheric data such as altitude, humidity, temperature, and pressure and be able to store that data into a database in the RPZ. To view that data the rocket must be recovered and the RPZ must be powered on through the whole flight of the rocket.

The objective was completed, and 7 launches were made where data was successfully recorded by the RPZ using the BME380 and the H3L. The data gathered was sufficient however it would have been better to have more launch force data however because of error in calculation it didn’t happen. Fixing the issues with time skipping for some of the database as well as fixing issues at code not running on startup after the 7th launch will need to be address for the next stages of this project. The future of this project is to get a rocket that would be able to reach higher altitudes and have a bigger payload as well as a more reliable source of power and better wiring.

1. Safety & Ethics

A rigorous safety protocol has been followed for when dealing with soldering, the construction of the rocket, and launching of the rocket. The rocket was only launched in the area assigned by the project description. When the rocket was launch there was an appropriate distance between the rocket and anything it could harm. Retrieving the rocket was also safe and caution was taken as to inspect the damage in case the lithium-ion battery had damage. The Ethics of this project fall in like with the IEEE code of Ethics.

1. Acknowledgements

Research Group:

Noah Laruent & Skylar Powell

Funding:

Funded by Texas Tech University ECE department supplying materials and lab spaces.

Equipment:

ECE Laboratory-116 Bench 117

Programs: Thonny, Maria Database

Technical Support & Rubrics:

Storrs, Written Reports for Project Lab at Texas Tech University

Texas Tech University, Undergraduate Laboratories Written Report Format

Mark haustein, General project direction and teaching instruction

1. References
2. Adafruit Industries, “Adafruit BME280 Humidity + Barometric Pressure + Temperature Sensor Breakout,” *https://cdn-learn.adafruit.com/downloads/pdf/adafruit-bme280-humidity-barometric-pressure-temperature-sensor-breakout.pdf*, [Accessed October 2022].
3. SparkFun Electronics, “PiZero\_1,” [*https://cdn.sparkfun.com/assets/learn\_tutorials/6/7/6/PiZerov2.png*](https://cdn.sparkfun.com/assets/learn_tutorials/6/7/6/PiZerov2.png), [Accessed October 2022].
4. eTechnophiles, “Raspberry Pi Zero GPIO pinout, specifications and programming language,”  July 2022 [Accessed October 2022].
5. SparkFun Electronics, “Sparkfun Triple Axis Accelerometer Breakout - H3LIS331DL,” SEN-14480, *https://www.sparkfun.com/products/14480,* [Accessed October 2022].
6. SparkFun Electronics, “MEMS motion sensor:  low-power high-g 3-axis digital accelerometer,” H3LIS331DL, [*https://cdn.sparkfun.com/assets/c/6/5/8/d/en.DM00053090.pdf*](https://cdn.sparkfun.com/assets/c/6/5/8/d/en.DM00053090.pdf), [Accessed October 2022].
7. IEEE, “IEEE code of Ethics,” *https://www.ieee.org/about/corporate/governance/p7-8.html*, [Accessed October 2022].
8. Table

   Description automatically generated Appendix

Appendix Budget

The budget sheet was created for the entire length of the project. At the end only around 110 hours per person was put into the project out of the 200 initially thought. The project was completed with about 50% of the original calculated budget. No contract labor was needed.

Graphical user interface

Description automatically generated Appendix GANTT CHART

The GANTT chart shows the completed timeline of the project and that all tasks were completed before the end date of the project.

Graphical user interface, application, table

Description automatically generated Appendix Material Cost

The material cost at the end of the project mounted to $564.50.

Appendix Code

from adafruit\_bme280 import basic as adafruit\_bme280

import board

import time

import datetime

import pymysql

i2c = board.I2C()

bme280 = adafruit\_bme280.Adafruit\_BME280\_I2C(i2c)

import smbus

import RPi.GPIO as GPIO

import time,os

from signal import signal, SIGINT, SIGTERM

from sys import exit

GPIO.setwarnings(False)

addr = 0x19

maxScale = 100

LED = 26

allData = open("AllSensorData.txt", "a")

alrtData = open("alertData.txt", "a")

CTRL\_REG1 = 0x20

CTRL\_REG4 = 0x23

OUT\_X\_L = 0x28

OUT\_X\_H = 0x29

OUT\_Y\_L = 0x2A

OUT\_Y\_H = 0x2B

OUT\_Z\_L = 0x2C

OUT\_Z\_H = 0x2D

POWERMODE\_NORMAL = 0x27

RANGE\_100G = 0x00

RANGE\_200G = 0x10

RANGE\_400G = 0x30

bus = smbus.SMBus(1)

GPIO.setmode(GPIO.BCM)

GPIO.setup(LED, GPIO.OUT)

GPIO.output(LED, GPIO.LOW)

def initialize(addr, maxScale):

    scale = int(maxScale)

    bus.write\_byte\_data(addr, CTRL\_REG1, POWERMODE\_NORMAL)

    if maxScale == 100:

        bus.write\_byte\_data(addr, CTRL\_REG4, RANGE\_100G)

    elif maxScale == 200:

        bus.write\_byte\_data(addr, CTRL\_REG4, RANGE\_200G)

    elif maxScale == 400:

        bus.write\_byte\_data(addr, CTRL\_REG4, RANGE\_400G)

    else:

        print("Error")

def readAxes(addr):

    data0 = bus.read\_byte\_data(addr, OUT\_X\_L)

    data1 = bus.read\_byte\_data(addr, OUT\_X\_H)

    data2 = bus.read\_byte\_data(addr, OUT\_Y\_L)

    data3 = bus.read\_byte\_data(addr, OUT\_Y\_H)

    data4 = bus.read\_byte\_data(addr, OUT\_Z\_L)

    data5 = bus.read\_byte\_data(addr, OUT\_Z\_H)

    x = data0 | data1 << 8

    y = data2 | data3 << 8

    z = data4 | data5 << 8

    if x > 32767:

        x -= 65536

    if y > 32767:

        y -= 65536

    if z > 32767:

        z -= 65536

    x = ~x

    y = ~y

    z = ~z

    return x,y,z

def convertToG(maxScale, xAcc1, yAcc1, zAcc1):

    X = (2\*float(maxScale) \* float(xAcc1))/(2\*\*16);

    Y = (2\*float(maxScale) \* float(yAcc1))/(2\*\*16);

    Z = -1\*((2\*float(maxScale) \* float(zAcc1))/(2\*\*16))-1;

    return X, Y, Z

def convertToA(maxScale, x, y, z):

    A = abs(x) \* 9.81

    B = abs(y) \* 9.81

    C = abs(z) \* 9.81

    return A, B, C

def cleanup(signal\_received, frame):

    allData.close()

    alrtData.close()

    GPIO.cleanup()

    exit(0)

def acc():

    signal(SIGINT, cleanup)

    signal(SIGTERM, cleanup)

    print("Starting Stream")

    initialize(addr,100)

    ts = time.ctime()

    allData.write(str(ts) + "\t")

    xAcc1, yAcc1, zAcc1 = readAxes(addr)

    x, y, z = convertToG(maxScale, xAcc1, yAcc1, zAcc1)

    i, h, u = convertToA(maxScale, x, y, z)

    fo = (0.146)\*((i + h + u)/3) #146 grams total rocket weight with payload

    allData.write("x: " + str(i) + "\t" + "y: " + str(h) + "\t" + "z: " + str(u) + "\n")

    if (i >= h and i >= u):

        return i \* 0.146

    if(h >= i and h >= u):

        return h \* 0.146

    if(u >= i and u >= h):

        return u \* 0.146

db = pymysql.connect(host="localhost",user="newuser",password="newuserpassword",database="RaspberryPi")

RD = """INSERT INTO ROCKETDATAC (date\_YYYY\_MM\_DD,time\_HH\_MM\_SS,temp\_C,humidity\_percent,pressure\_hPa,altitude\_Meters,launch\_force\_N) values(%s,%s,%s,%s,%s,%s,%s);"""

cursor = db.cursor()

while True:

    bme280.sea\_level\_pressure = 1025.096 # value changes depending on day

    pascals = (int)(bme280.pressure + (bme280.altitude / 8.3))

    degrees = (int)(bme280.temperature)

    humidity = (int)(bme280.relative\_humidity)

    altitude = (int)(bme280.altitude)

    csvt = datetime.datetime.now()

    today = datetime.date.today()

    date = datetime.date.isoformat(today)

    tim = datetime.datetime.now()

    t = tim.strftime("%H:%M")

    force = acc()

    cursor.execute(RD, (date, t, degrees, humidity, pascals, altitude, force))

    db.commit()