Code Documentation

Table of contents

Early planning and Goals

Starting this very ambitious project the code was meant to do 2 things log the data and approximate an firing angle for the launcher. As a team we decided that this goal was going to be auxiliary to producing and testing the launched projectile as time showed that even the basics required work forcing us to remove some of the scope of the project. In tern the team shifted goals to first completing the projectile rather than the launcher and an integrated firing mechanism.

Personal Goals of the project

After the scope of the project was shifted to the projectile one of the main goals for me Paul as the programmer was to try new things and push the code and me into learning. The approach I took to this was to incorporate an advanced system for approximating the position of the projectile and to heavily incorporate mathematical modeling into the project as I had some previous experience doing that writing a mechanum wheel drivetrain simulator and wanted to keep doing that.

Code overview

The goal of the project was to separate the code into 2 distinct parts

1. Data Logging
2. Data Analysis
   These run in different parts because one is circuit python and one is regular Python. This allows one program to use more complex library's such as numpy or SCIPY while the other runs all the hardware specific code.

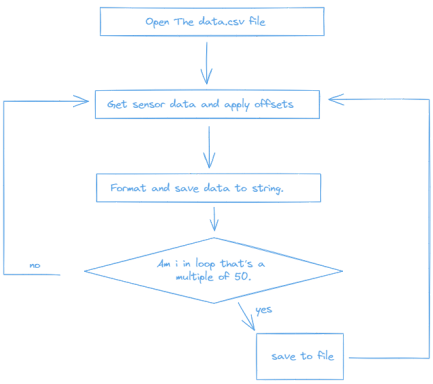
Data Logging

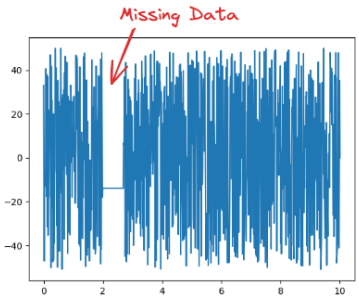
The data logging program is exactly as it sounds it logs the value from the LSMDOX(IMU) and the barometer allows it to long height, all axis acceleration variables and time. Data logging also had specific requirements as to make sure that the data would be accurate and useful as to be used later.

1. Always collect data at relatively even intervals. (this is why time is stored)
2. Contain data that is accurate at the required polling rate.
3. Store the data on an onboard file.

Polling/writing

Because the onboard storage is relatively large the original goal was to write to an onboard CSV file after every 50 data points were collected. This although turned out to be impossible as the act of writing data to the CSV file would create "gaps" or holes as when a file was being written it wouldn't be able to poll the sensors. This is a big problem when collecting data as it means that i can only store variables in the heap (as a variable) which offers much less space than the on board storage (264kB vs 2mb). This forced me to reduce the polling rate as i calculated that i could only store about 10,000 points of data reducing the amount of storage by an order of magnitude. This limitation was adapted around as the final solution to the problem was to store all of the points in ram but to erase any old data after the 10,000 had been collected and to keep doing so until the program loop ended and then that current "slice" of data was written to file.





Dead Reckoning

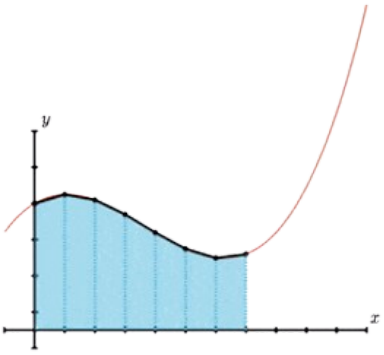
As stated previously one of the wished goals of this project was to push myself and dead reckoning was the way to do it, dead reckoning is a technique for approximating the positing of an object using its previous positions. In this context this means taking the acceleration values out of the IMU and using those to approximate the real position of the SMORT in flight.

Translating Acceleration to Position

Translating acceleration to position is pretty straight forward as all we have to do is sum all of the previous acceleration values to get the velocity and then sum them again to get the position. Mathematically we can just take the integral using a numeric method.

$$P(t) = \int \int a(t)\, dt\, dt$$

The best numerical method in this case is trapezoidal integration due to the fact that we can represent the space between 2 points as trapezoids and that this represents no loss in accuracy from the data to the integral as the method is just as accurate as the points.



$$= \frac{y_1 + y_2}{2} \Delta x + \frac{y_2 + y_3}{2} \Delta x + \ldots$$

The way that this is done in code is using the formula above which in reality is just the formula for a trapezoid $\frac{a_0+a_1}{2} h$ repeated. This is then implemented in code through the line 5 and 6.

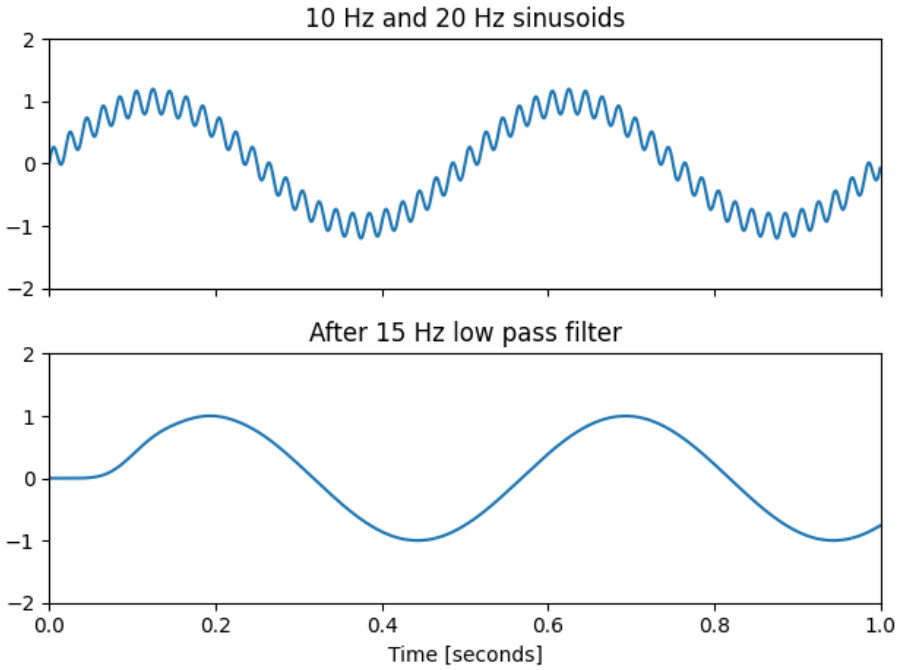
```
def dbl_cumtrapz(x,y):
    r_array = np.zeros(len(x))
    for i in range(1, len(x)):
        r_array[i] =   y[i-1] +  (y[i-1] + y[i])/2 * (x[i] - x[i-1])
    r_array[0] = r_array[1] # normalize the first value
    return r_array
```

Data filtering

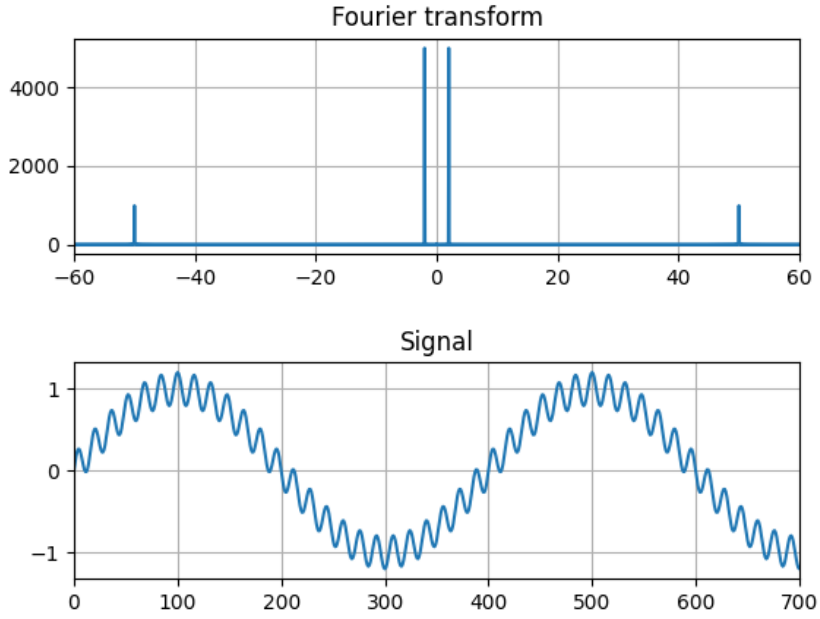
Data filtering or more specifically signal processing was where the most time was "wasted" as Cyrus would say as it deem to be the most difficult and trouble some although that was in part intended. The goal of this project was to use a **discrete second order low pass Butterworth filter.** to smooth out the acceleration data to reduce the compounding error.

What is a Low pass filter

A low pass filter is a mathematical algorithm and often physical circuit that removes the high frequency part of a certain function and keeps the low frequency part hence the name low pass. Each of these filter functions has a cutoff frequency which in a reductive sense removes this frequency from the data.



To find out what cutoff frequency to use take the Fourier transform of the data which basically approximates a mathematical equation for the data using a sum of sinusoidal waves. Each of these sinusoidal waves has a certain frequency and by separating them out you can intern find out what frequency the error is as it will be represented as one or many different $\sin$ waves of a similar frequency.



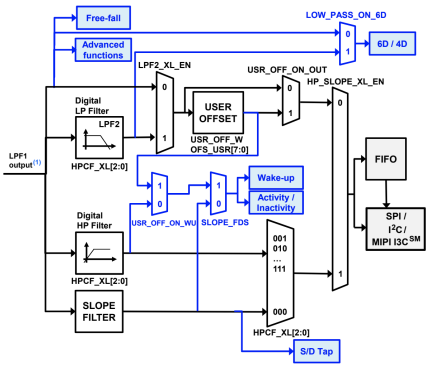
> Note that both of these graphs are just examples and not sample data from an IMU.

Why Signal Processing Failed

Signal processing is a complicated and very advanced mathematical realm for someone who was taking calculus 1 at the time. I did learn alot about it and my knowledge was fair and good enough to set up the code to do what i wanted it to do but not really good enough to debug and go much farther beyond what was written in the API documentation. This bogged me down spending hours trying to debug stuff where it was really hard to parse solutions and also was extremely frustrating as often i just had no idea how to approach a problem.

Embedded register configuration

Embedded register configuration is when you change the CPU registers on embedded devices through a certain serial interface usually through I^2C . In my case the LSM6DOX had a cup register which would have allowed me to use built in low pass filtering on the actual hardware removing my need for any external signal processing and only requiring me to integrate the result which in theory I could just do on the pico anyways.

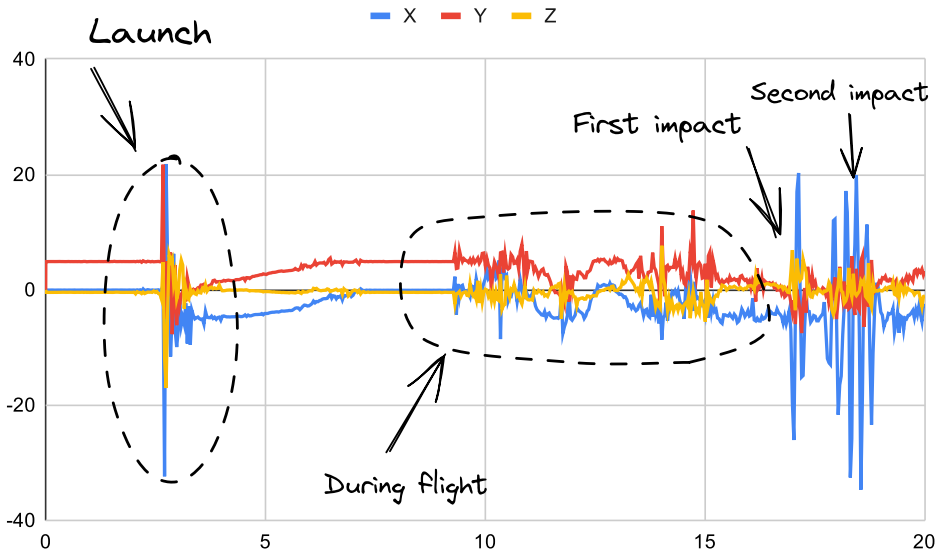


> Accelerometer block diagram (if anyone understands what this means contact me 🙏)

Post launch Results

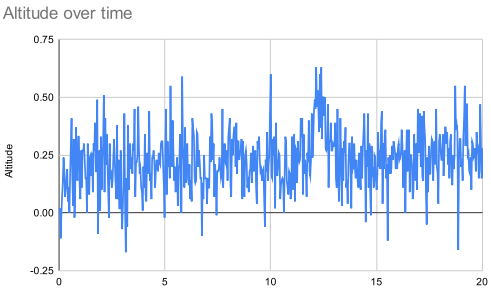
The team only ended up having 1 opportunity to do a launch with data collection. This limitation was a product of a lengthy data logging code development cycle.

Sadly data was only collected on the first launch presumptively because the damage of the first launch caused damage extensive enough to stop logging (see below). This although doesn't discount the results of the first launch as although the cycle rate of the data wasn't high enough to effectively use the Dead Reckoning aspect of the project.



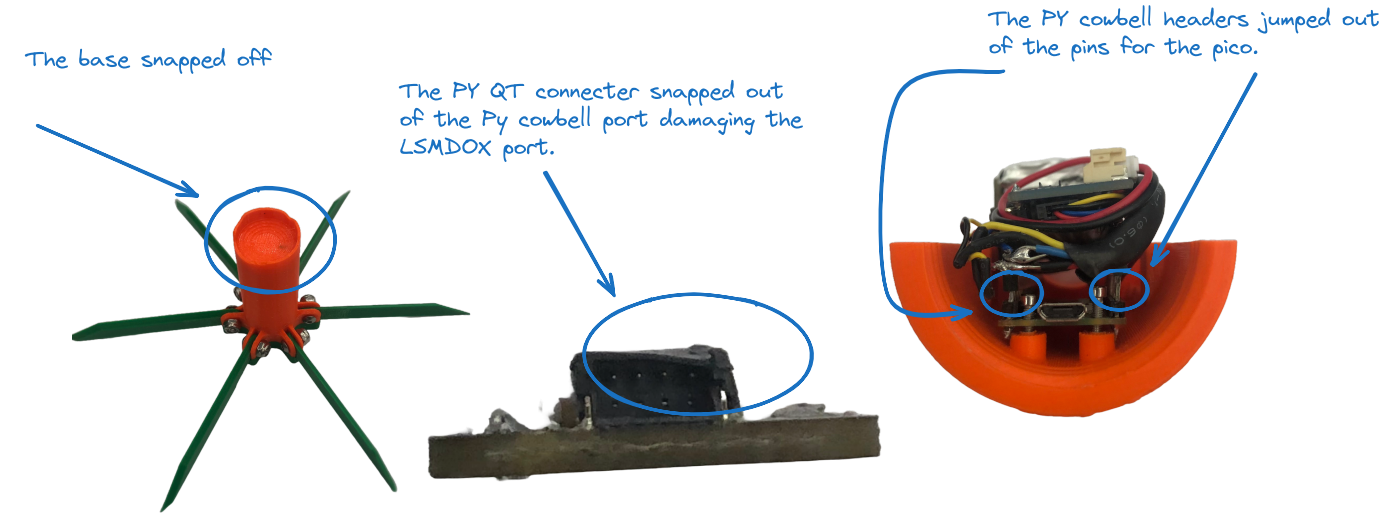
Altitude data Issues

sadly the altitude data wasn't accurate and showed no variation, basically saying that the projectile didn't move at all as seen in the range. Although we dont really know why this happened it most likely was the fact that the inside of the projectile was pressurized in the sense that the barometer on the Altimeter couldnt detect the outside pressure because the pressure wasn't changing.



Mechanical Failures

The main data collection launches were done twice in consecutively and at first seemed to go well as on the first launch we had no hardware break as the spring seemed to do its job of taking the brunt of the impact although the tail did snap off (bottom left image below). The second launch however didn't go as planned as the SMORT started to tumble due to the fact that it no longer had stabilizing fins thus causing it to take the brunt of the impact on insides damaging the electronics as seen in the images below.

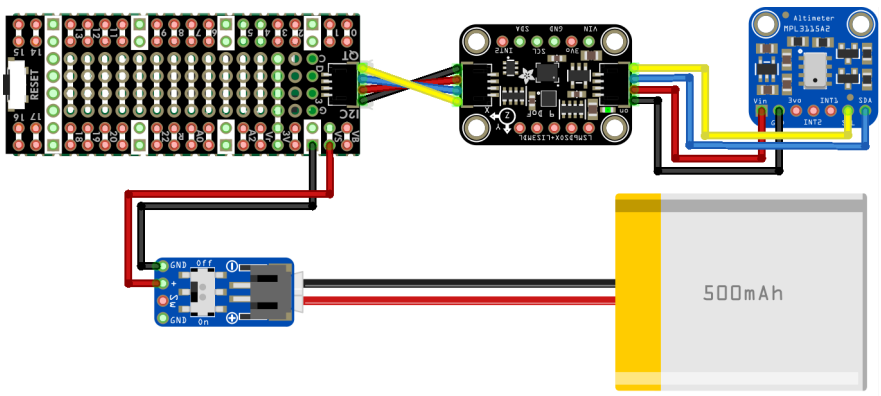


Damage after launch

Wiring

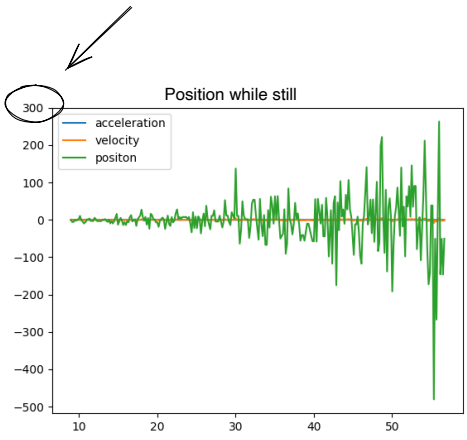
Wiring Design

The main goals of the wiring design where to be easy to assemble and compact. We had to shift from the traditional breadboard design for this exact reason as we didn't have the space to fit such a large surface. Another factor is the fact that we didn't need any complex wiring as we aren't using control surfaces or anything that requires large batteries or electronics this means we theoretically just need a power source and an I^2C connection which means we only need 4 wires per component. This 4 wire system Taylors perfectly to the Stemma QT wires and headers as they contain the only the wires we need. This also creates a problem as the stemma QT headers aren't found on the Pico and normally using something like a feather would have been more practical. The solution was to use a PY cowbell a prototyping shield which has the Stemma header and for us functions basically as an adapter. With the components sorted all we need to do is power the board which can be done using the lipo battery and attaching a JST header to the cowbell.



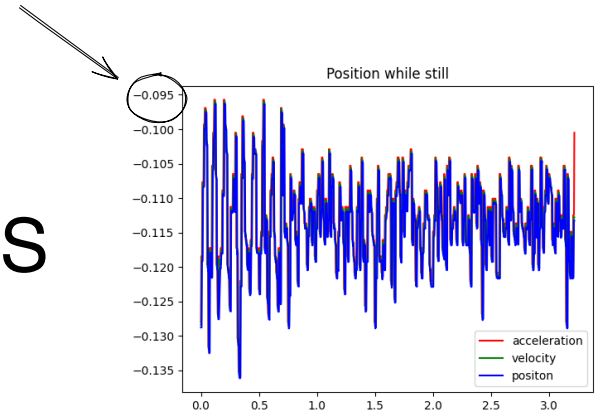
IMU choice

This project started with an MPU 6050 but this IMU revealed itself to be too inaccurate to be used for Dead Reckoning as the data even when still was so inaccurate that it would be hundreds of meters off. Therefore I switched over to the LSM6DSOX as it was supposed to be more accurate and less error prone. This switch turned out to be a good choice as the LSM6DSOX was much more accurate.



MPU 6050

VS



LSM6DSOX

> Take this data with a grain of salt as the LSM6DSOX data might be inaccurate due to method of integration.

Bonus meme Mr. Miller often says that the astronauts who landed on the moon had less computing power than us and still managed land on the moon I mean yea but that's like saying Einstein didn't have a calculator I am no Einstein trust me.