UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

ECE 445 - Design Document

Extending IMU Degrees of Freedom for Pose Estimation Using AI on Chip

Spring 2023

Team 71
Chirag Rastogi (chiragr2)
Lukas Zscherpel (lukasez2)

1. Introduction

1.1 Problem

An Inertial measurement unit (IMU) is a combination of sensors that collects data based on movement. IMU's normally include an accelerometer and a gyroscope which track the specific acceleration and the angular acceleration of the object.

The sensors are:

- Accelerometers: Used to measure linear acceleration in three dimensions. This information can be used to estimate the velocity and position of the object over time.
- Gyroscopes: Used to measure angular velocity in three dimensions. This information can be used to estimate the orientation of the object over time.
- Magnetometers: Used to measure the direction of the Earth's magnetic field. This information can be used to determine the orientation of the object with respect to the Earth's magnetic field, which can be used to correct errors in the orientation estimate obtained from the gyroscopes.

IMU's are used in a wide range of applications but they are really important in the medical field and in consumer electronics. Some example applications include movement tracking on patients to detect disorders or even tracking movement in your cell phone to get its orientation.
9DOF IMU sensors can be found for as low as \$10-\$20 for basic models, but these sensors have lower accuracy. For projects that require greater accuracy, the cost can go up to 300\$ (https://x-io.co.uk/ngimu/) and this limits projects that require multiple such devices.

1.2 Solution

An AI on chip solution may have the potential to reduce the cost of 9DOF IMU sensors by enabling the integration of multiple sensors and processing functions onto a single chip, which can simplify the design, reduce the bill of materials, and lower the manufacturing costs.

By leveraging AI algorithms among others, an AI on chip can enable 9DOF IMU sensors to perform advanced sensing and processing tasks on-device, reducing the data transmission requirements and minimizing the need for external computing resources.

Our solution is to take a cheap 6 DOF IMU and combine it with a RNN that we train to calculate the other 3 DOF that a magnetometer normally provides. We will then take this AI model and put it onto a chip. The AI on chip will work together with the 6DOF IMU to emulate a 9 DOF IMU in a handheld format.

1.3 Visual Aid

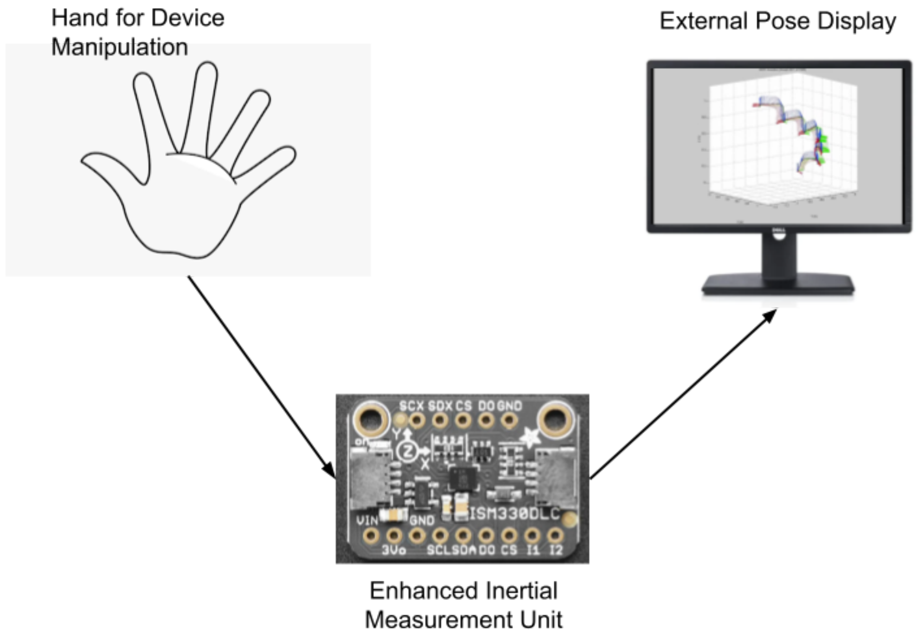


Figure 1. Visual Aid

1.4 High-Level Requirements List

- The IMU interfacing system must be able to accurately measure and monitor the orientation and movement of a device to within a maximum error of 3 degrees and 0.5 cm/s.
- The system must be able to perform calibration on the data outputted by the IMU to within a maximum error of 0.5 degrees and 0.2 cm/s to ensure accuracy and consistency.
- For a given IMU the system must be able to determine the parameters for each algorithm, out of a set of predefined algorithms, that are effective at minimizing the error and noise outputted by the IMU. This would ensure a Mean absolute percentage error of less than 5% over 20 mins of data collection.

2. Design

2.1 Block Diagram

The proposed project comprises of two block diagrams, each representing a different stage of the project. The first block diagram depicts the initial deliverable, which is intended to serve as the minimum viable product for the project. This block diagram focuses on testing various algorithms on the NVidia Jetson, which will be used to process the data generated by the IMU. The separation of the Jetson from the rest of the subsystems enables us to quickly and easily deploy different algorithms without having to make any changes to the other subsystems.

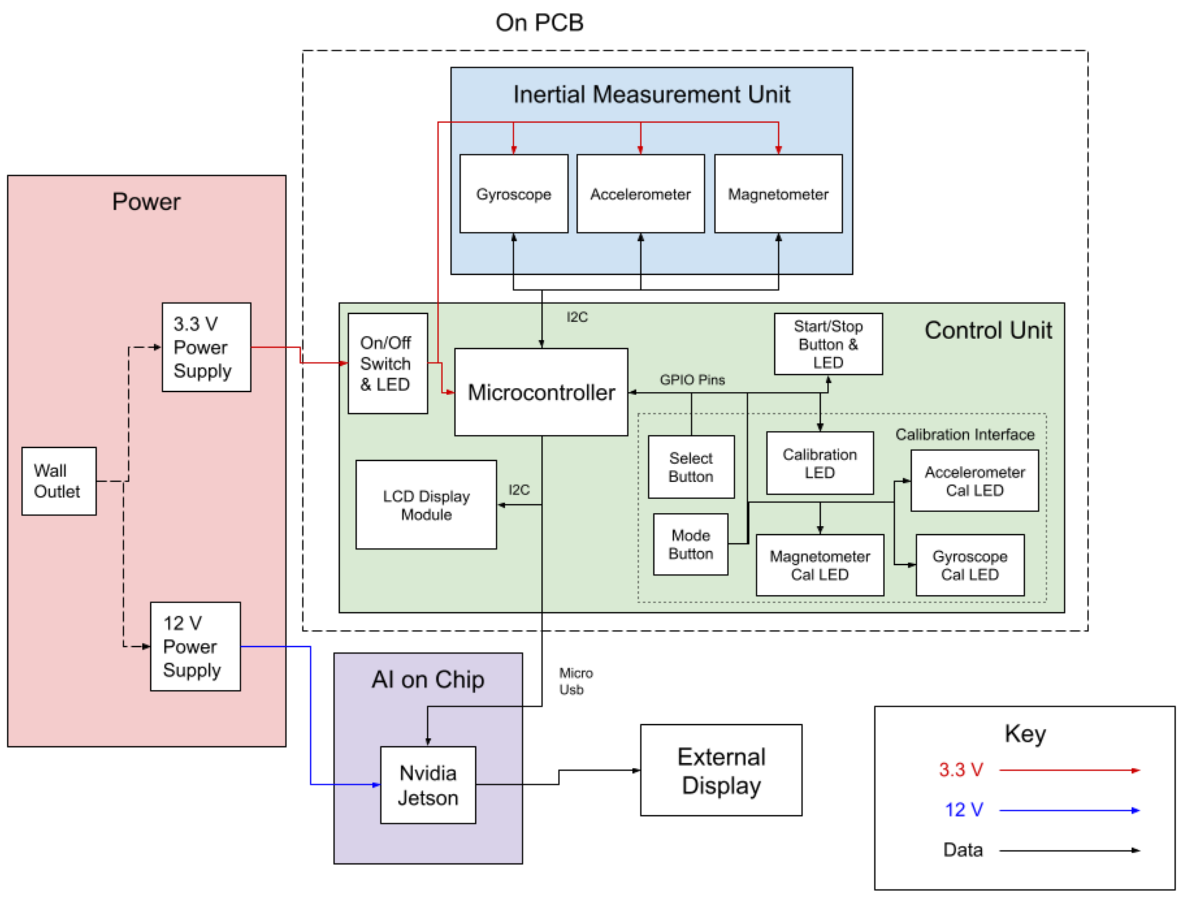


Figure 2. Initial Deliverable Block Diagram

The second block diagram represents the reach goal of the project. After evaluating the performance of various algorithms, we aim to integrate the most promising ones into the PCB. This could be accomplished through hardware implementations using filters or onto a FPGA unit. In this design, the data processing will be directly connected to the IMU, eliminating the need for an intermediary control unit. However, the control unit will still play a critical role in managing the flow of data and providing the same functionalities as in the initial deliverable design.

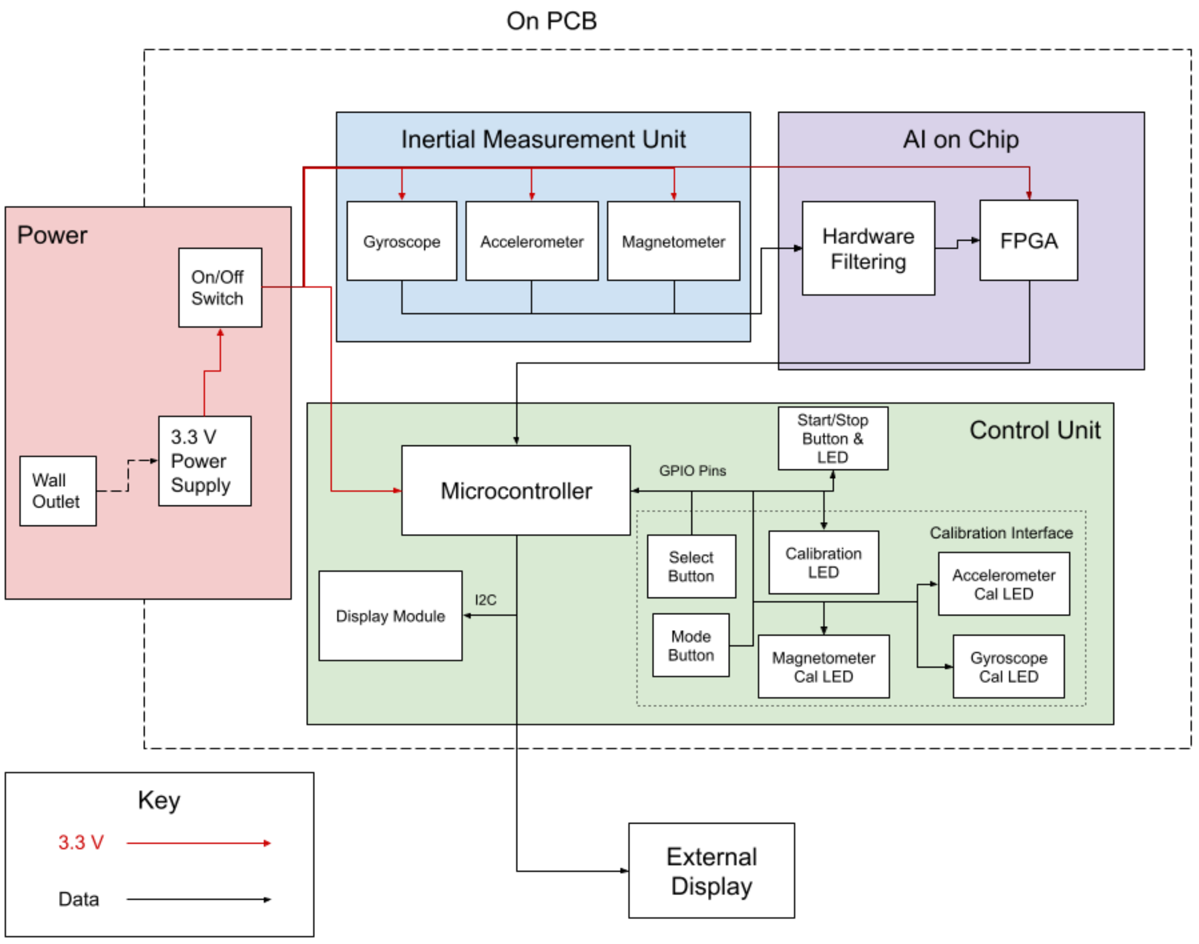


Figure 3. Reach Design with Hardware Accelerated AI Block Diagram

2.2 Subsystem 1: Inertial Measurement Unit

This subsystem will consist of a 6 DOF or 9 DOF IMU that we acquire from a third party distributor. We plan on using a MPU-6050 and MPU-9250 for this project. We will test the accuracy of these two IMU's and how well they work when combined with noise-reducing algorithms to produce the most accurate data with the least amount of noise. The MPU-6050 and MPU-9250 are relatively similar IMU's with a couple key differences. They both have onboard temperature sensors which can be used when calibrating the data output. They also both contain a Digital Motion Processor (DMP) which correlates the data from the sensors. The differences between the two IMU's is that the MPU-9250 has 9 DOF while the MPU-6050 has 6 DOF. The MPU-9250 also claims to have more accuracy within its sensors when compared to the MPU-6050. While both IMU's are relatively low cost, inside the United States the MPU-9250 can be found for a price ranging from around $7 to $20 while the MPU-6050 can be found for a price ranging from around $4 to $10.

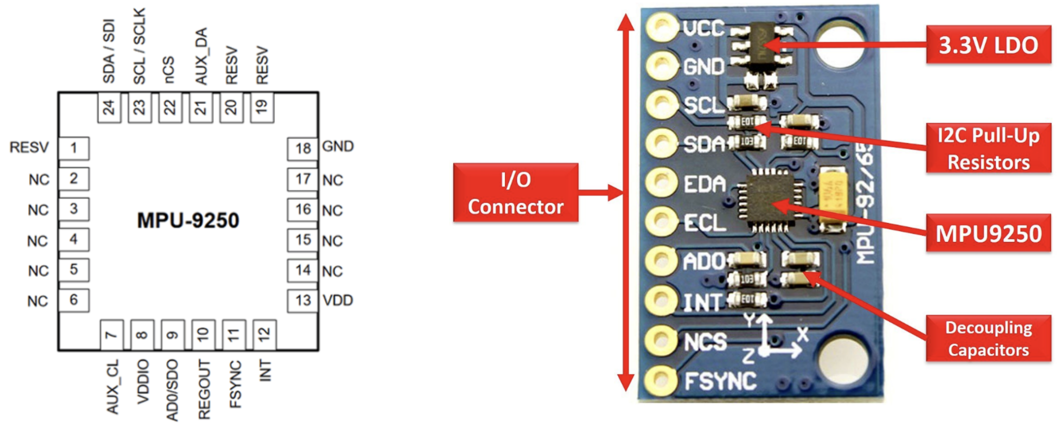


Figure 4: MPU-9250 chip pinout and board model

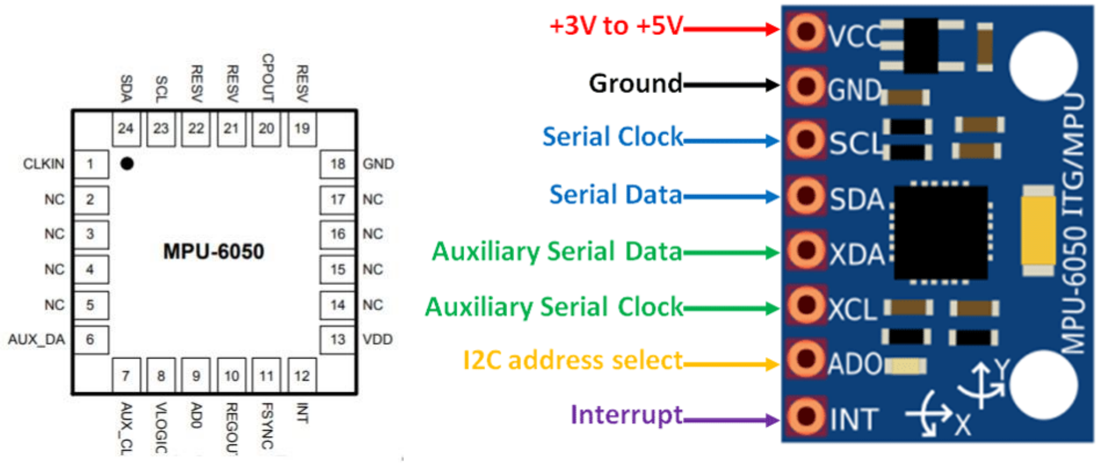


Figure 5: MPU-6050 chip pinout and board model

Both IMU's require very little power. They both run on 3.3 volts, but they accept an input voltage in a range between 3 and 5 volts as they contain internal voltage regulator systems. Both IMU's also draw very little current and require no more than 5mA of current which can be easily supplied by the power supply subsystem.

We will connect this IMU to the microcontroller to get values from the accelerometer, gyroscope and the magnetometer if available. This connection will be done through the I2C protocol as it is supported by both IMU's as well as the microcontroller. The IMU's don't have a built in way of calibrating their sensors, so the calibration and application of offset will be done by the microcontroller. The IMU will simply pass on the raw data it collects from its sensors. The IMU's do have multiple different power modes which we will be utilizing during the calibration and testing processes. The different modes can be seen in the following table (For the MPU 6050 there are less modes available as it does not have an internal gyroscope).

Mode	**Name**	**Gyro**	**Accel**	**Magnetometer**	**DMP**
1	Sleep Mode	Off	Off	Off	Off
2	Standby Mode	Drive On	Off	Off	Off
3	Low-Power Accelerometer Mode	Off	Duty-Cycled	Off	On or Off
4	Low-Noise Accelerometer Mode	Off	On	Off	On or Off
5	Gyroscope Mode	On	Off	Off	On or Off
6	Magnetometer Mode	Off	Off	On	On or Off
7	Accel + Gyro Mode	On	On	Off	On or Off
8	Accel + Magnetometer Mode	Off	On	On	On or Off
9	9-Axis Mode	On	On	On	On or Off

We are able to specify which of the modes to use using commands in the I2C protocol and we plan on testing the use cases for each of the modes.

Requirement	**Verification**
● The IMU must accurately measure orientation and motion data with a minimal error rate within the manufacturer's tolerance range.	● The error rate of the IMU for measuring orientation data must be less than 3 degrees within the manufacturer's tolerance range of ±2 degrees.
● The IMU must operate at the same voltage level as the rest of the PCB and communicate using the I2C protocol with a low error rate.	● The IMU must operate at 3.3V with a tolerance of 0.2V ● The I2C communication between the IMU and the microcontroller must have an error rate of less than 1% for reliable data transfer.

• The IMU must have a stable and consistent output rate of at least 100Hz	• The output rate of the IMU must be within ±5Hz of the specified rate of 100Hz. • The output rate of the IMU must not fluctuate by more than ±2Hz over a 10-minute testing period.

2.3 Subsystem 2: Control Unit

The Control Unit is a critical component of our design, tasked with interfacing with the IMU and directing the raw data it outputs. Communication between the Control Unit and IMU will be established using the I2C data protocol, and the Control Unit will also be responsible for calibrating the IMU. Additionally, the microcontroller will be connected to a USB port for interfacing with the NVidia Jetson in the deliverable model and for communication with an external display in the reach model. The Control Unit, as the central entity for data flow management, enables us to incorporate user interface components such as a button for activating and deactivating data flow, buttons for calibrating the IMU to determine the error size, and a small OLED display screen to convey instructions and information to the user.

The control unit subsystem will be comprised of an ESP32 as the microcontroller, a 128x64 OLED display screen, 6 LED's of various colors, 3 push buttons, one 2 position toggle switch, a MCP2200 UART to USB chip, and a USB interface. The ESP32 has two I2C bus modules, one of which will be used to communicate with the IMU subsystem and the other to control the OLED display. The ESP32 also has 36 general purpose I/O pins (GPIO) which is plenty to connect with any switches, buttons or LED's that we will use. The ESP32 has 520kB of SRAM and 2MB of on board flash memory which is plenty for storing the code for the program and data that we collect temporarily during the calibration process. The RP2040 also runs at 160MHz which is fast enough to collect the data from the IMU without any losses. The 128x64 OLED display screen is a small cheap monochrome OLED screen that can display graphics and text. The 128x64 OLED display has a built in I2C module which makes it simple to connect to the ESP32, it is also supported by many open source code libraries which makes programming the ESP32 to use the display straightforward. We plan to use the display to communicate with the user what mode the mode the microcontroller is in, to display the data when the microcontroller is transmitting data from the IMU, and to give instructions to the user when the microcontroller is calibrating the IMU.

Connected to one of the ESP32's GPIO pins will be a power on/off switch which will control whether the control unit and IMU subsystems will be powered on or off. This switch will be connected to the 3V3_EN port of the ESP32. In conjunction with the power switch will be an LED which is turned on when power to the rest of the PCB is enabled. This helps give visual information to the user letting them know that the ESP32 has finished booting up and is ready to run. We will use one of the push buttons to control whether the microcontroller is in data transmission mode, this push button will be the start/stop button labeled in the block diagram. During the data transmission mode the microcontroller will collect data from all the sensors on the IMU, apply the offsets it has calculated during the IMU calibration, then send the data straight out of the USB port to the connected device. If this push button is pressed when the microcontroller is calibrating the IMU the input on the button will be ignored until the calibration has finished and the user has chosen to exit calibration mode. There will also be a LED that is used in conjunction with the transmission mode button that will light up to let the use know that data is being transmitted.

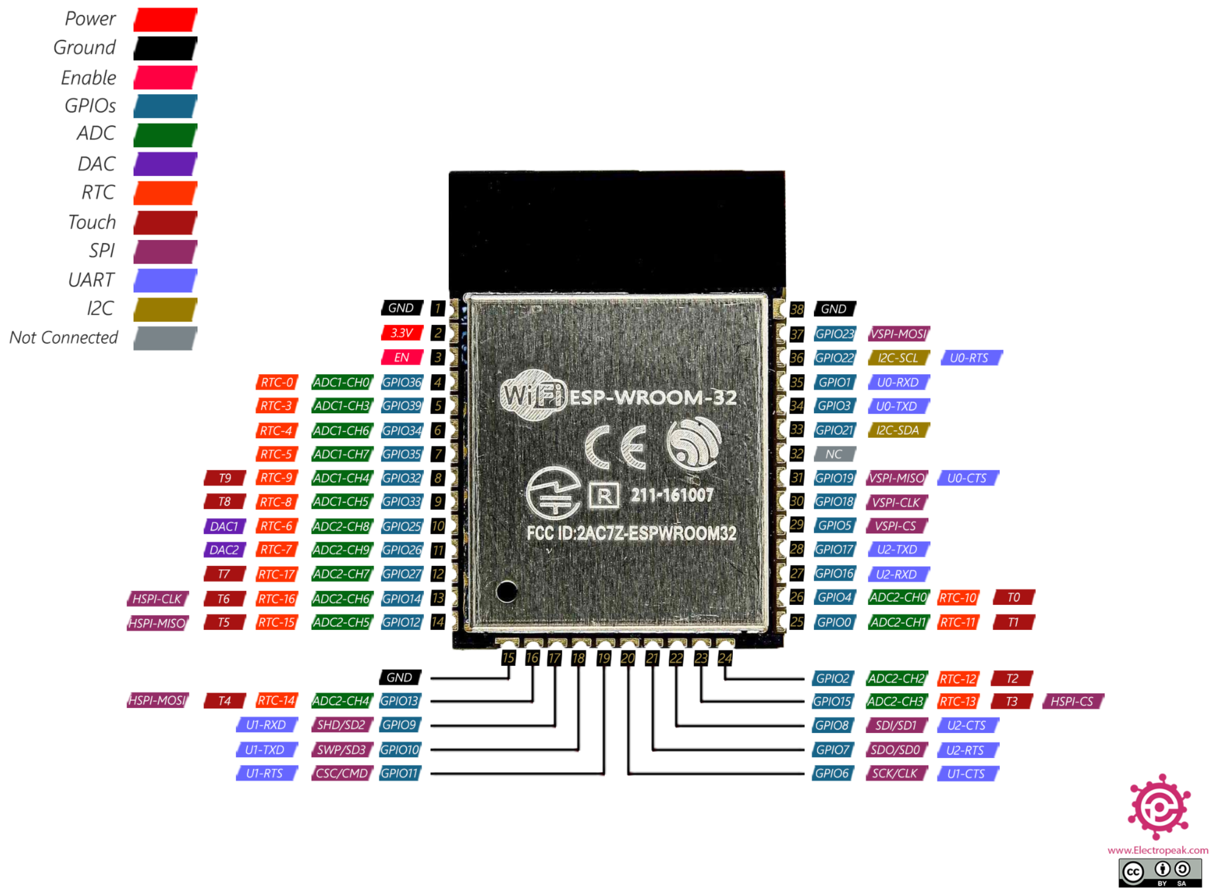
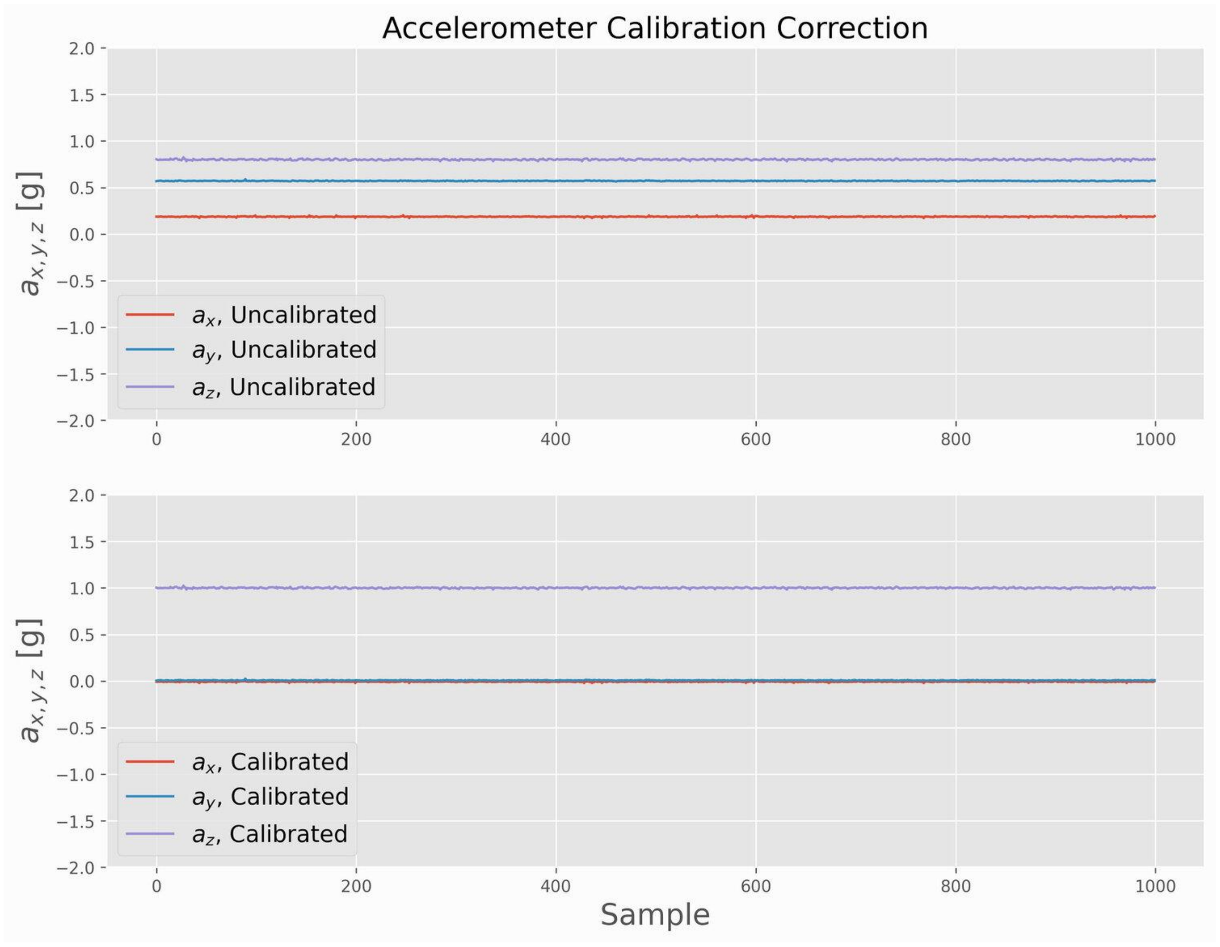


Figure 6:ESP 32 pinouts

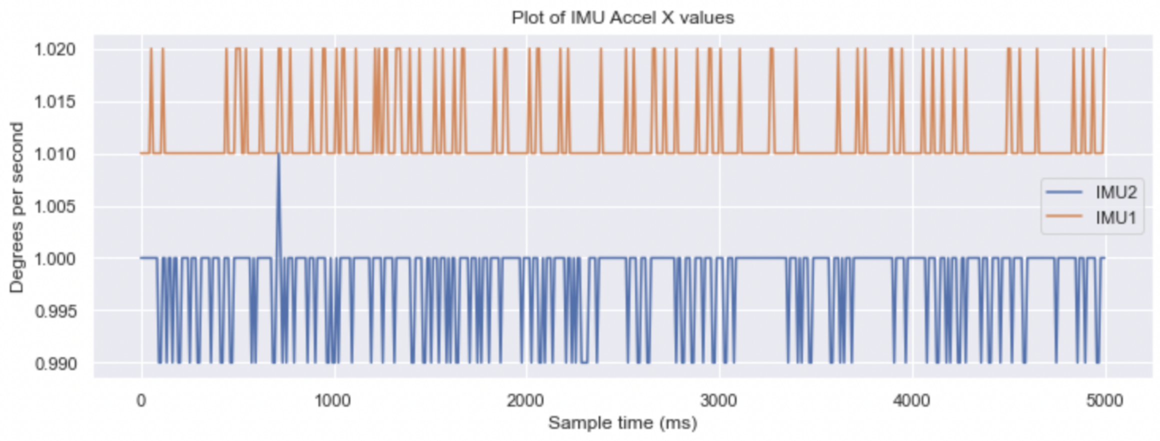
The rest of the buttons and LEDs will be exclusively used to interface with the user when the user is trying to calibrate the IMU. These buttons and LEDs are the calibration select button, the calibration mode button, the calibration LED, the accelerometer calibration LED, the gyroscope calibration LED, and the magnetometer calibration LED. If the microcontroller is powered on and the microcontroller is not in the data transmission mode, the user will be able to press the calibration select button to enter the calibration mode. Once the microcontroller has entered the calibration mode then the calibration LED will turn on letting the user know that they have successfully entered the mode. The microcontroller will then display options on the OLED display for the user to select either accelerometer, gyroscope, or magnetometer calibration or to exit the calibration mode. To change the selections between the different options the user can press the calibrate mode button. In addition to the OLED display highlighting the choice the user is currently selecting, each of the respective sensor's LEDs will blink when that mode is highlighted and the calibration LED will blink if the user is highlighting the exit calibration mode option. Once the user has highlighted the choice they want to select they can press the calibration select button to confirm that choice. To help with the calibration of the IMU the PCB with the IMU will be attached to a 3D printed cube.

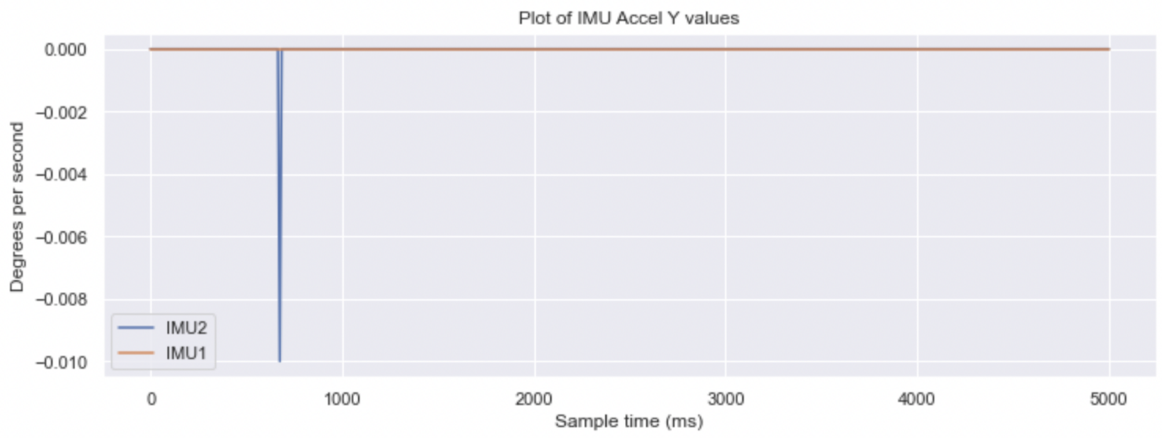
If the user has selected to calibrate the accelerometer then the microcontroller will enter the accelerometer calibration mode and the accelerometer calibration LED will be turned on. To calibrate the accelerometer we will first implement a simple method using gravity, if we are able to calibrate the accelerometer successfully using this method we will then attempt to validate the calibration using numerical integration and forces other than gravity. The simple method of calibration using gravity is as follows: we will have the user put the PCB with the IMU onto a level surface and have them wait for a couple seconds so that we can record the accelerometers readings for each DOF. After the microcontroller has finished recording the values for that position, we will have the user turn the IMU onto a different face of the cube and let the microcontroller record data again. This will be repeated for each of the 6 faces of the cube so that each DOF that the accelerometer reads is able to experience a force of 1g, -1g, and 0g. After all the data has been recorded the microcontroller will then calculate the average offset of the data from the expected value at each of the different rotations to create an offset matrix.

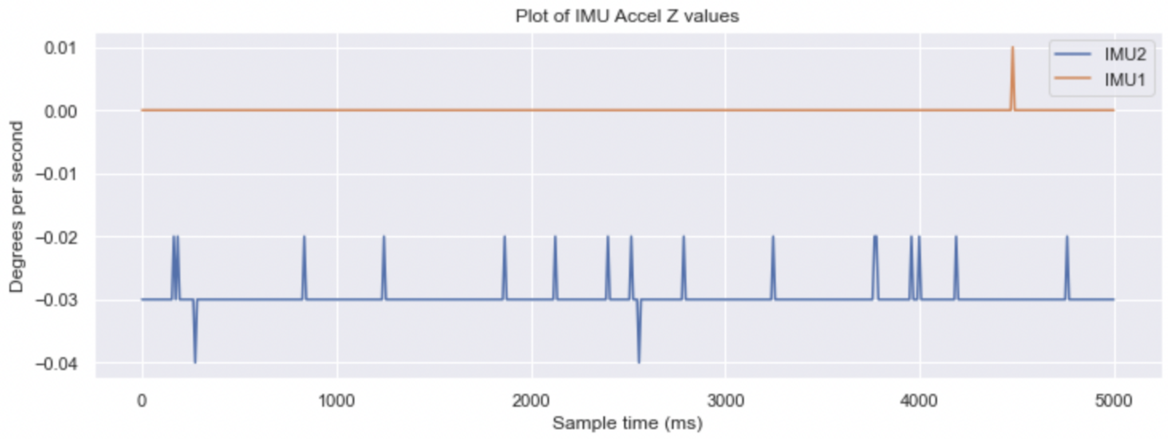
Figure 7: Accelerometer data uncalibrated vs calibrated example



The following are plots are created using data.







If the user has selected to calibrate the gyroscope then the microcontroller will enter the gyroscope calibration mode and the gyroscope calibration LED will be turned on. There are many different ways to calibrate the gyroscope, but for our initial deliverable we will implement a simple calibration routine. If we are able to successfully implement the simple calibration routine we will explore validating the calibration through numerical integration as well as other calibration routines such as using the onboard temperature sensor for the IMU. The basic gyroscope calibration routine is as follows: we will simply have the user place the IMU onto a flat surface and record the data that the gyroscope sensor reads. Because the IMU is just sitting still and not rotating, the values read by the gyroscope should be centered around zero, if the data read by the gyroscope is not equal to zero we will calculate the offset of the mean of each DOF and store it within a matrix to be applied later.

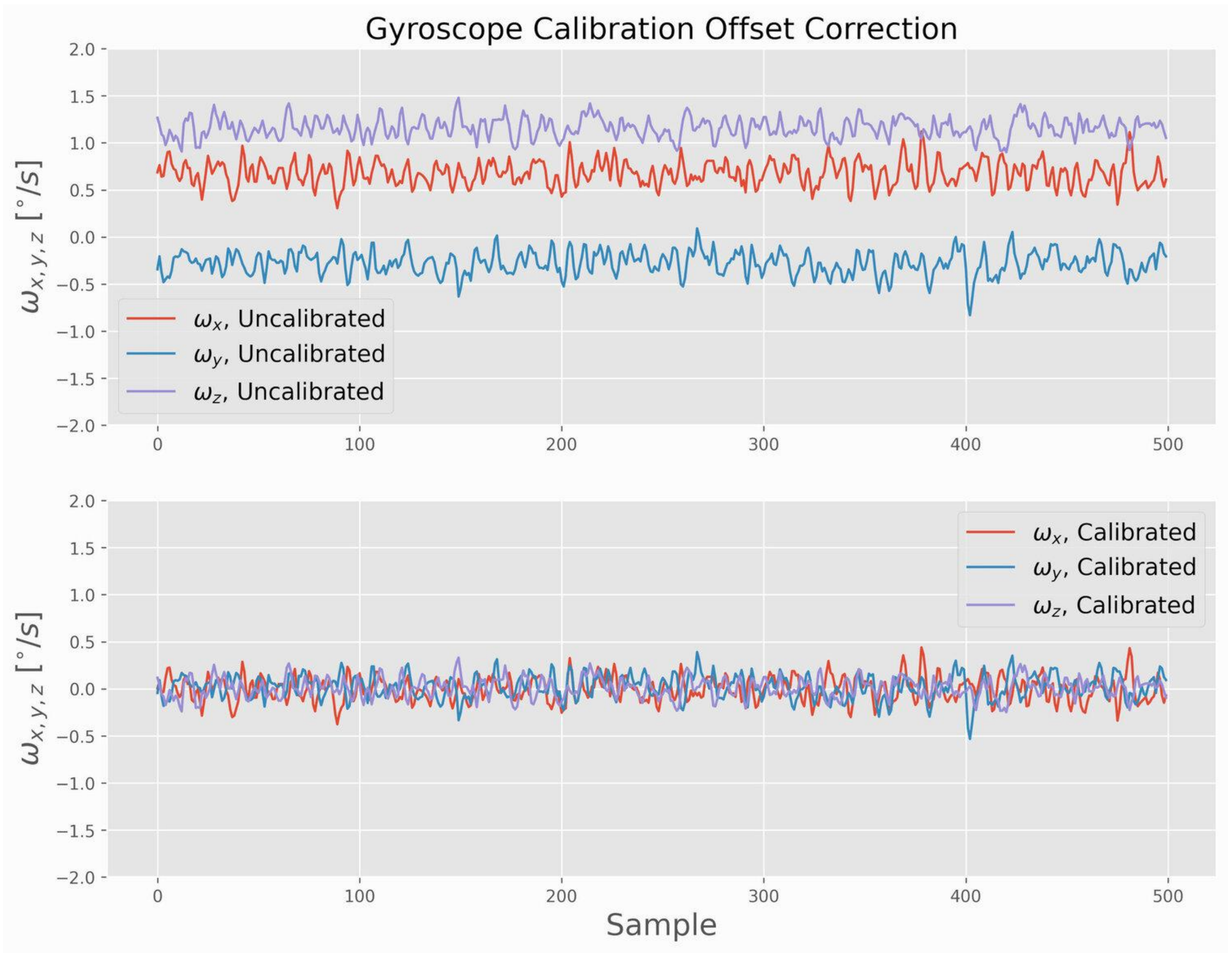
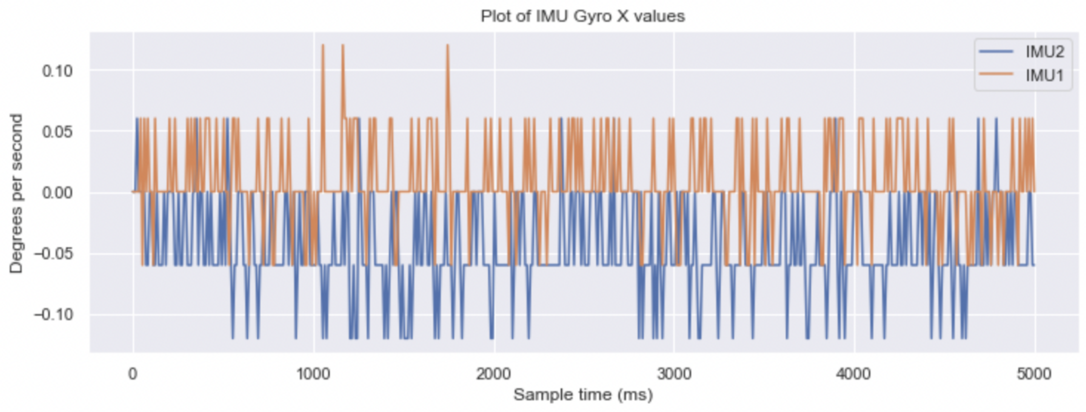
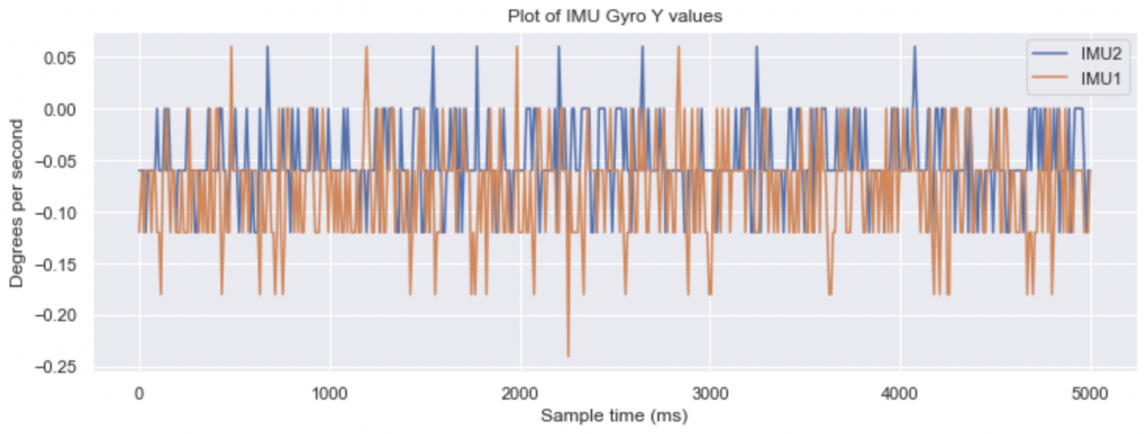
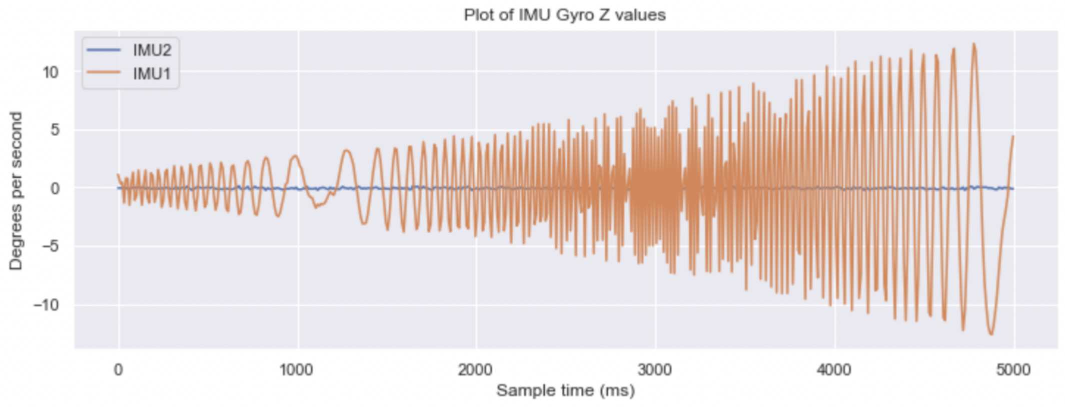


Figure 8: Gyroscope data uncalibrated vs calibrated example

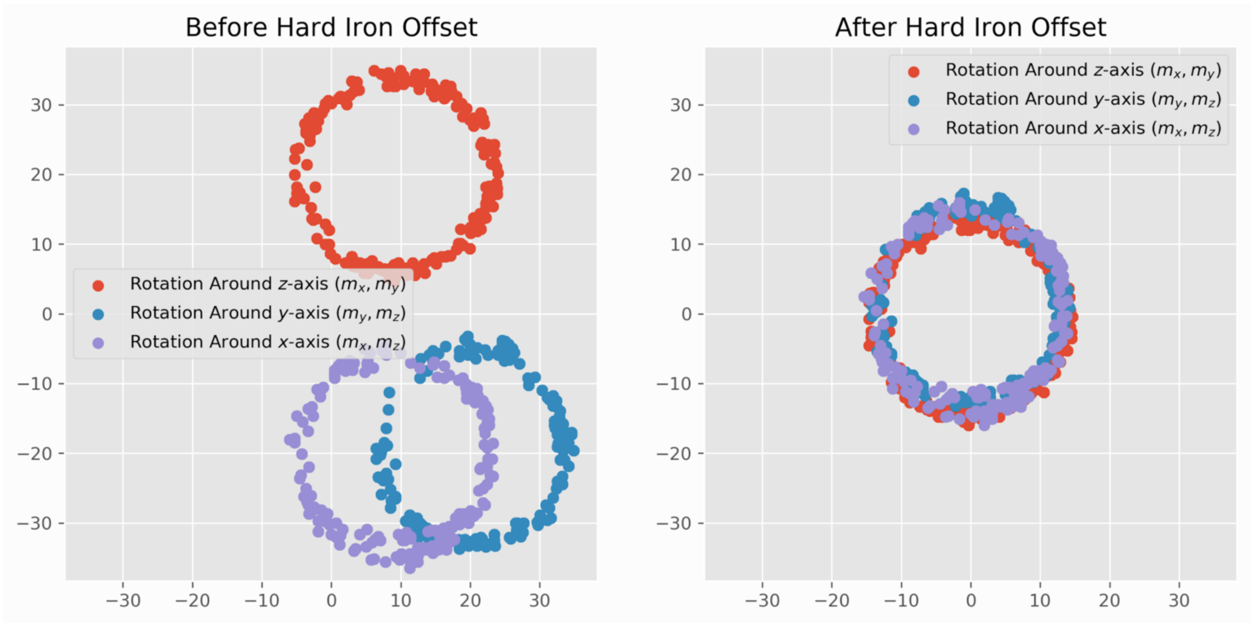
The following are plots are created using data.







If the user has selected to calibrate the magnetometer then the microcontroller will enter the magnetometer calibration mode and the magnetometer calibration LED will be turned on. When calibrating a magnetometer there are two different calibrations that can be performed depending on the data that is recorded by the magnetometer. These calibration methods are hard iron offsets and soft iron offsets. For the initial deliverable of this project we will focus on performing a hard iron offset to the data as soft iron offsets are more complicated and are only necessary if the rotation of the IMU gyroscope data does not maintain a circular form. Hard iron offset calibrations are performed as follows: for each axis of the IMU we will have the user rotate the block 360 degrees around that access multiple times for a defined amount of time, during this time the microcontroller will record the magnetic response of the planar magnetometer sensors. If graphed the data will be represented by a circular figure which denotes the rotation around the relative axis for each planar magnetometer sensor. We calibrate this data by calculating the offset of the center of each circle from the origin of the graph. Each of these calculated values represent the hard iron offset for each axis.



In order to streamline the calibration and make it easy for any user to perform an instruction manual will be created with information on how to calibrate each of the different sensors.

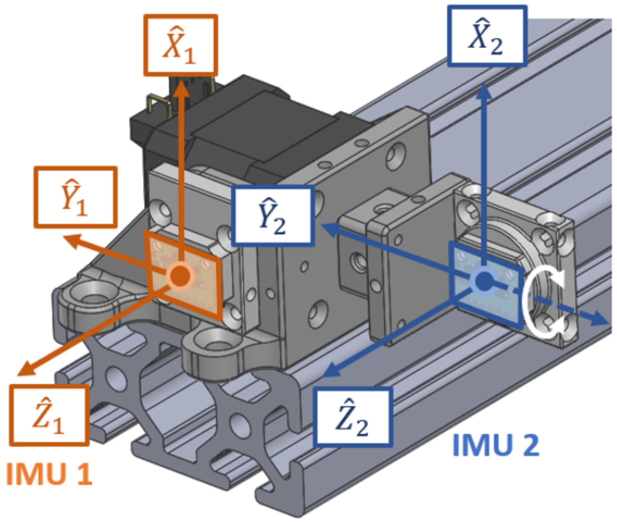
The calibration interface will be hosted on the PCB, but the processing of data for the calibration will be done on the Nvidia Jetson in the reach model of the project. This is because it allows us to use the PyCal library to process the data and make our jobs easier.

Requirement	**Verification**
● The Control Unit must be able to process and transmit data quickly and accurately from the IMU to the device connected to the USB port.	● The Control Unit must be able to process IMU data at a rate of at least 100Hz. ● The Control Unit must be able to transmit data to the device connected to the usb port at a rate of at least 10Mbps. ● The I2C communication link between the IMU and the ESP32 must have a maximum data transmission error rate of 5%.
● The control unit must perform accurate calibration of the raw IMU output data	● The calibrated IMU data must remain stable with less than less than 1% drift (pose and position) when the

to account for any manufacturing bias and environmental factors. In the reach model the calibration will be done on the Nvidia jetson.	IMU is placed on a flat surface with no rotation, for a duration of 20 minutes ● Verify that the calibrated data has a mean value of zero within a tolerance range of +/- 0.1 degrees ● Verify that the calibration process can be repeated with consistent results within a tolerance range of +/- 0.1 degrees. ● The calibration process must be able to calibrate each of the 3 different sensors and the data that they output. ● Verify that the calibration process does not introduce any additional errors or noise to the IMU's output data, and that the calibrated data accurately represents the physical orientation and motion of the sensor.
● The user interface of the Control Unit must be designed in such a way that it is easy for a user to perform calibration and data collection tasks without the need for extensive technical knowledge. The user should be able to initiate calibration and data collection processes with minimal effort and without the need for complex configuration or setup procedures.	● The user interface must have clearly labeled buttons or controls that correspond to each functionality, such as "Calibrate" and "Collect Data". Each button or control must be labeled with clear and concise text or symbols that are easily understood by the user. ● The user interface must provide clear instructions or prompts on how to initiate the calibration and data collection processes. These instructions or prompts should be displayed prominently on the interface and should be written in simple and easy-to-understand language. The user should be able to follow these instructions without any confusion or difficulty.

2.4 Subsystem 3: Position Estimation using AI on Chip

AI on chip either through Nvidia Jetson or fpga that will take the output of the IMU and predict what the orientation of the device will be.
We have deployed an LSTM model to begin with, however we will be looking into the following algorithms as well: Accelerometer Inclination, Gyroscopic Integration, Complementary Filter, Kalman Filter, Digital Motion Processing, Madgwick Filter, Mahony Filter. The hardest challenge of this project will be the hardware acceleration of these algorithms and processing data while dealing with noise.
https://ieee-dataport.org/open-access/estimating-relative-angle-between-two-6-axis-inertial-measurement-units-imus.



The data we will be using for this project is collected from 2 IMUs as shown above. The IMUs used are MPU-6050.

Here, IMU 1 does not move, while IMU 2 rotates about the Y Axis. IMU2 is attached to a motor and we use the encoder values as ground truth. 9 trials were conducted and each trial lasted 25 minutes. In this section, we go over the data and the algorithms used.

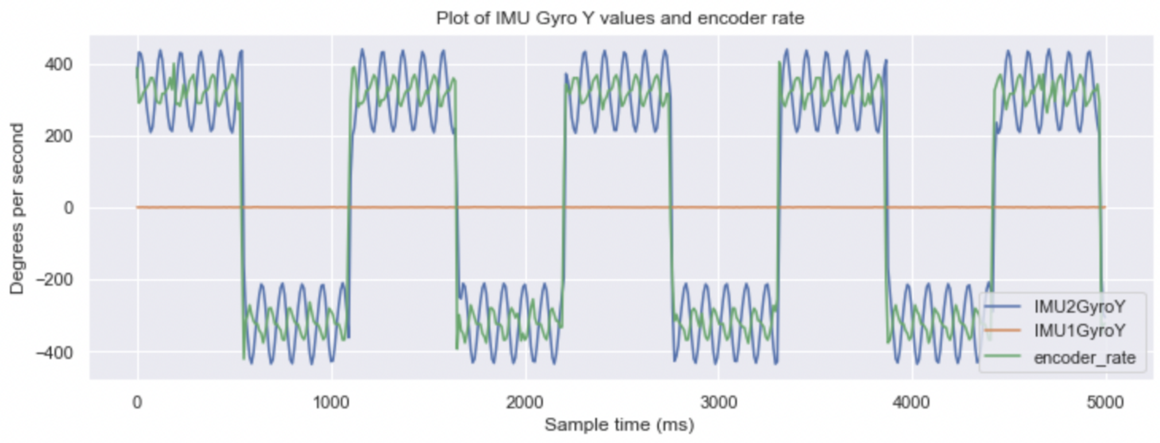
Gyroscope Values:

For the plots, we have compared the gyro reading across each axis on each IMU and placed them against the encoder rate. This encoder rate was obtained by dividing the encoder difference, i.e difference in degrees between 2 consecutive readings and the time difference (ideally 10ms but calculated based on time stamp).

Y axis

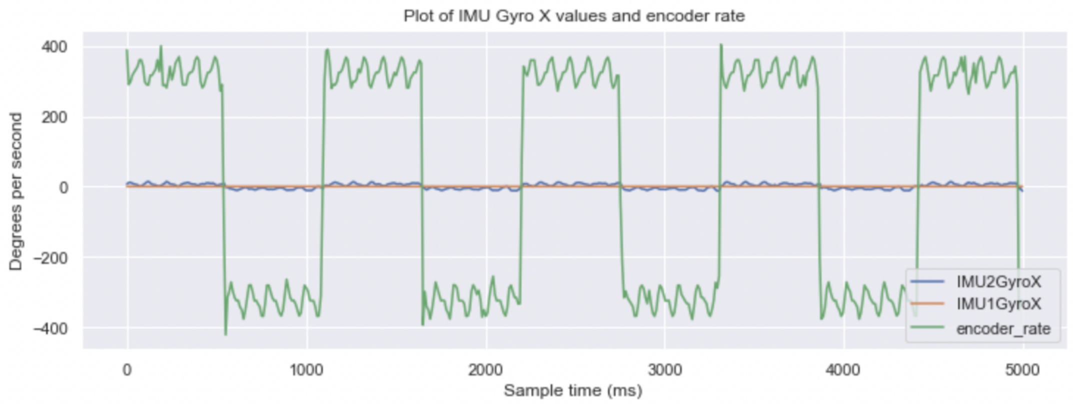
We begin with the Y axis readings as that is the only axis we are rotating about. As we can see, the Gyroscope value closely matches the encoder rate. This is a very good sign as it shows that the gyroscope readings even from the cheapest IMU can provide relatively accurate measurements. IMU1 gives a 0 reading with noise, and this is expected as well.

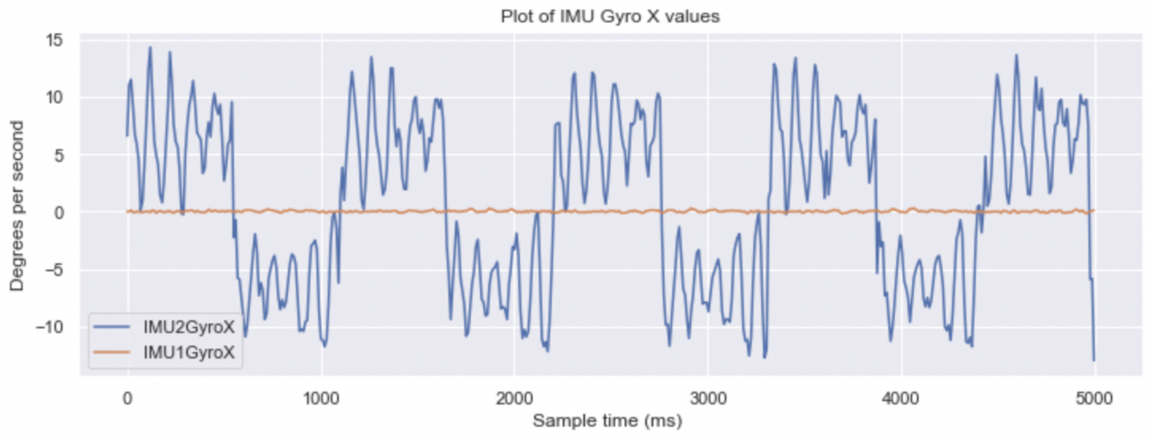
As we can see, the peaks of the Gyro are higher than that of the encoder rate. Here, we suspect that either the IMU was subjected to noise or the imu was not perfectly aligned when conducting the experiment. This is further supported by the next graphs.



X Axis

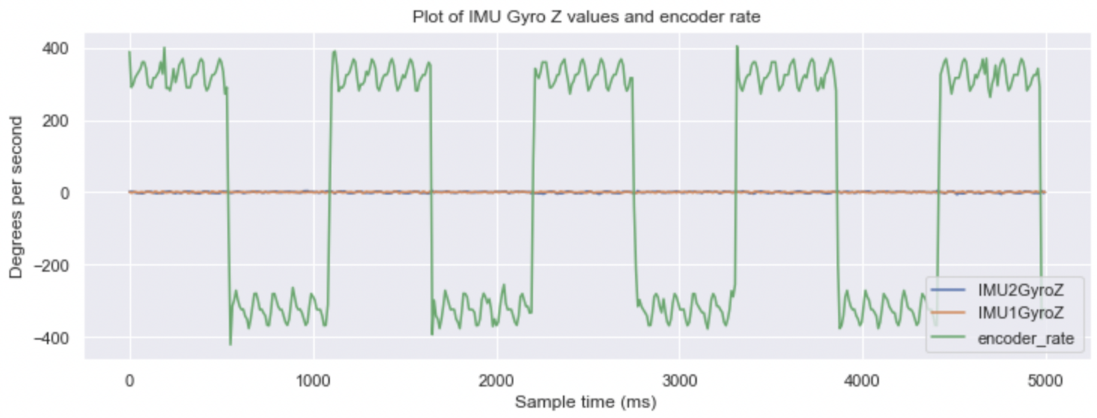
As we can see, the gyro readings are almost 0 for both. However, IMU 2 has non-zero readings that follow the pattern of the encoder. This leads us to believe that the IMU was either subjected to noise, or it was slightly tilted, causing values to be registered on the X axis as well. We will be reaching out to the lab that conducted this experiment to confirm. If it was perfectly aligned, we will find a way to accommodate this during calibration.

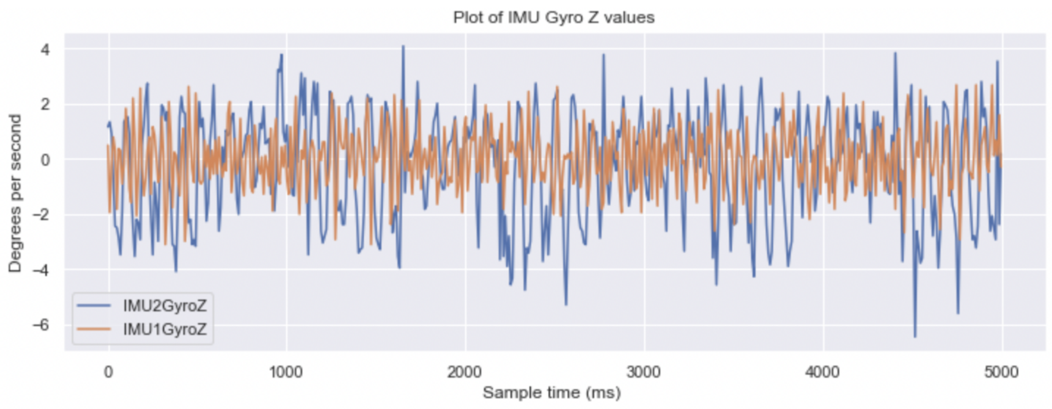




Z Axis

As we can see, the gyro readings are almost 0 for both and they are just jubjected to noise. IMU2 closely follows IMU1, showing that the rotation adds only upto a degree in noise.

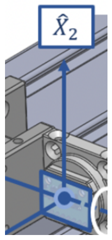




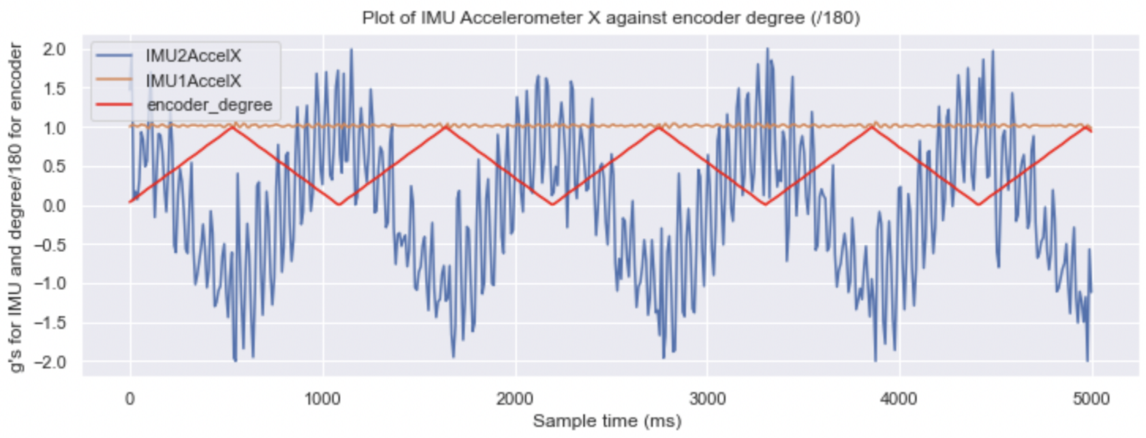
Accelerometer Values:

For the plots, we have compared the accelerometer reading across each axis on each IMU and placed them against the encoder degrees.

X Axis

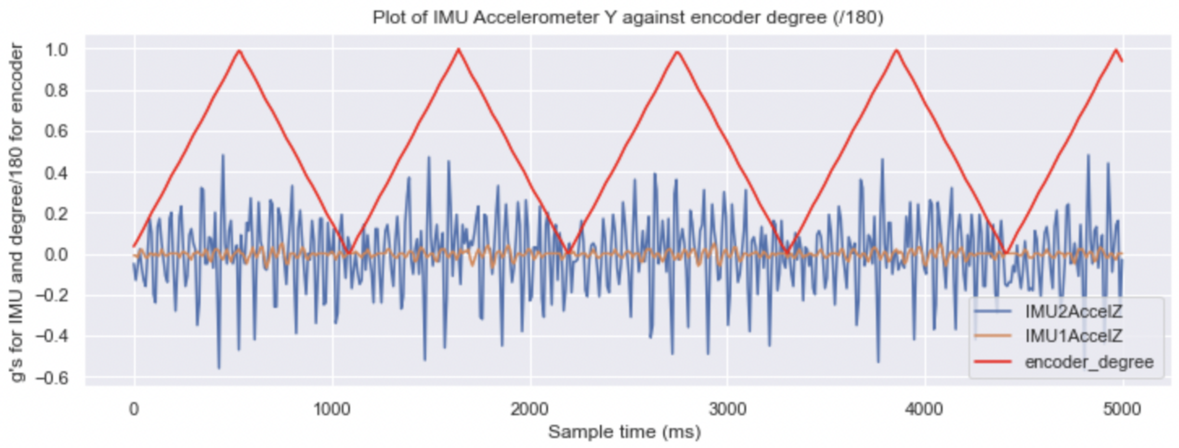


We expect the accelerometer to register periodic values on the X and Z axis as those are subject to movement. As we can see the X value begins with 2g as expected, as it experiences 1g from gravity at rest and 1g due to the speed. We see a periodic pattern with a lot of noise being repeated. This is something we can work on improving, as the noise causes a lot of errors to accumulate when calculating the 3d orientation. As we can see, the values accurately follow the encoder degree.



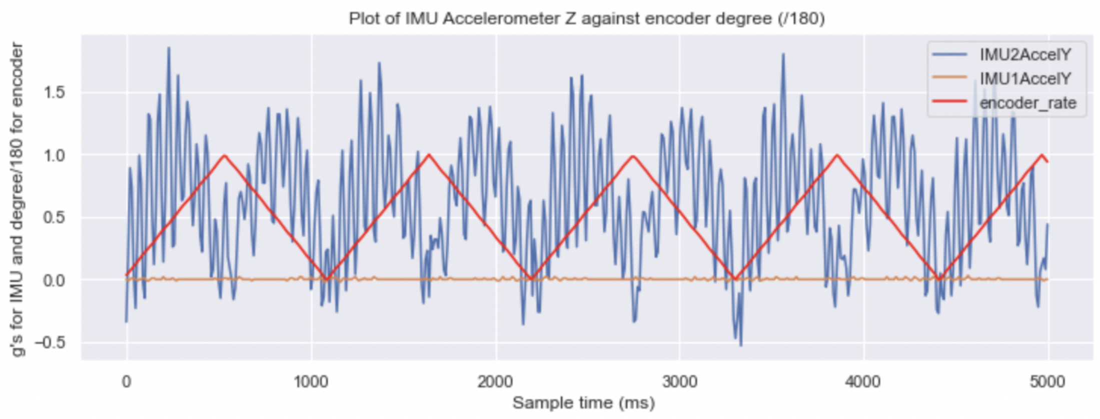
Y Axis

The Y axis is almost 0 however it follows the peaks and troughs of the encoder. This could be because of the tilt issue we mentioned earlier.

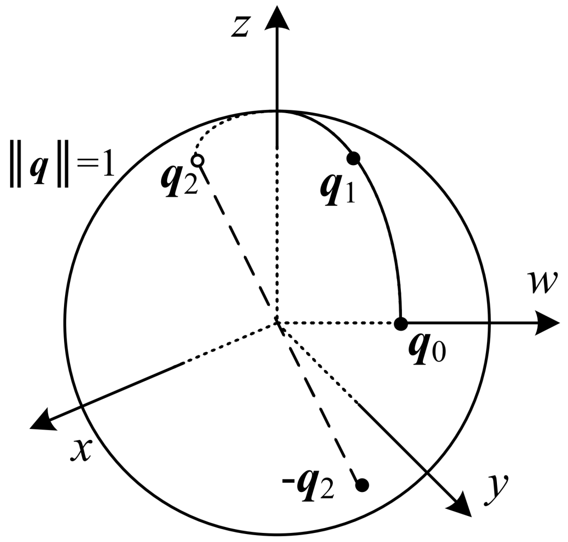


Z Axis

The Z axis follows the peaks and troughs of the encoder. As we can see, the values accurately follow the encoder degree.

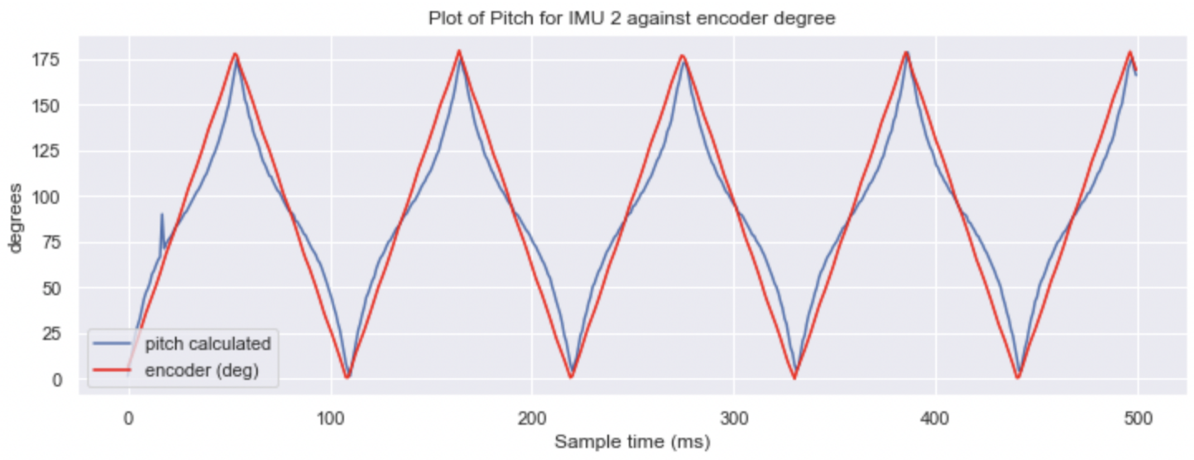


3D rotation from DMP

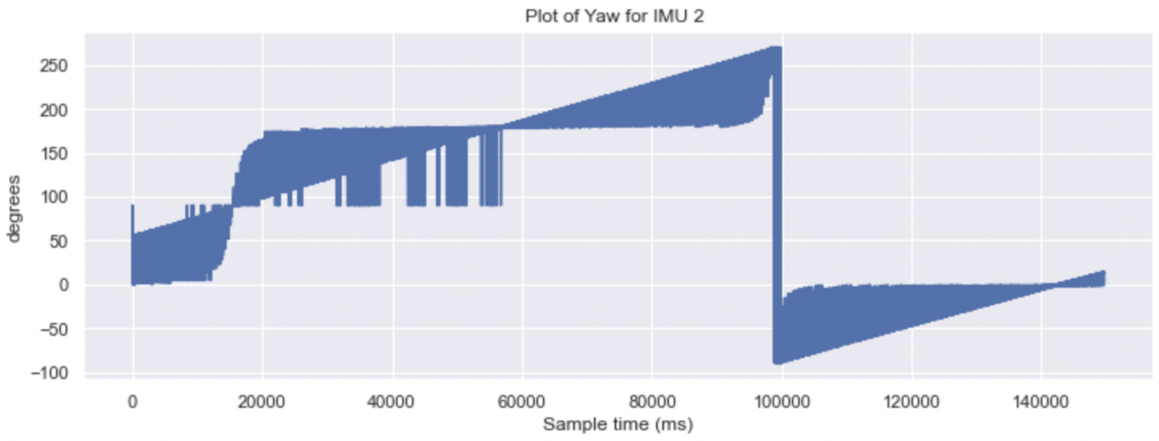


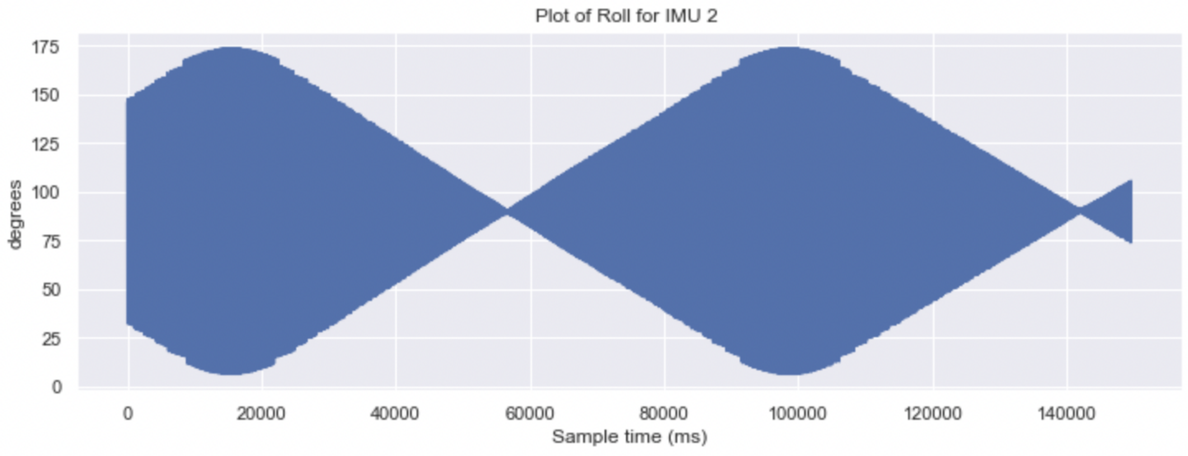
Quaternions for 3D orientation

We use the quaternion values calculated by the Digital Motion Processor (given by the author of the dataset) to obtain the pitch angle. As we can see, the pitch closely matches the pattern, with the peaks and troughs matching, however we believe that Deep learning algorithms can give a better estimate.



Seen below are the Yaw and Roll calculated for the IMU and as we can see, the Yaw is extremely inaccurate with the values drifting upward.

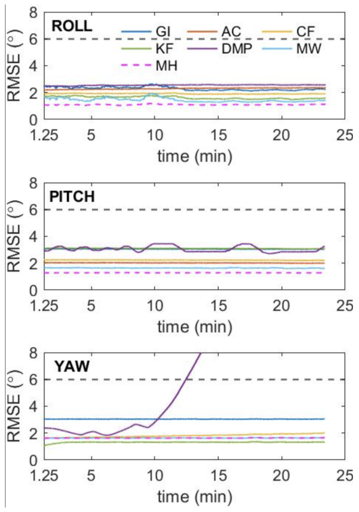




Standard Algorithms:

We will begin by implementing 3 of the 7 standard algorithms selected based on the performance seen below.

We will be implementing the Kalman Filter, Madgwick Filter, and Mahony Filter.

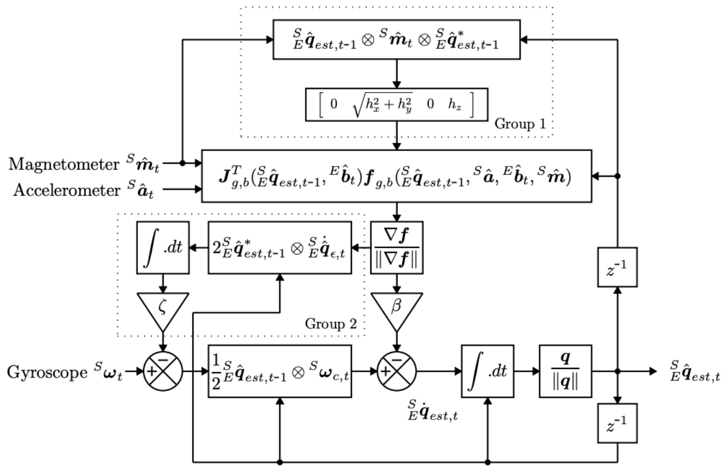


Kalman Filter:

A Kalman filter is a type of mathematical algorithm that is used to estimate the state of a system over time. In the context of orientation tracking, a Kalman filter can be used to estimate the orientation of an object by combining the measurements from an accelerometer and a gyroscope.

Madgwick Filter:

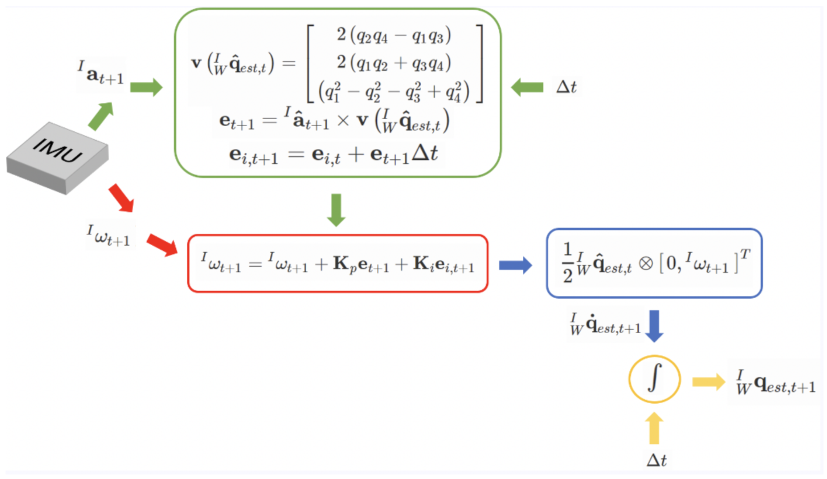
The Madgwick filter is a type of complementary filter used to estimate the orientation of an object.


Group 1
Group 2
Magnetometer $^S\hat{m}_t$
Accelerometer $^S\hat{a}_t$
Gyroscope $^S\omega_t$
${}^S_E\hat{q}_{est,t-1} \otimes {}^S\hat{m}_t \otimes {}^S_E\hat{q}^*_{est,t-1}$
$[\,0 \;\; \sqrt{h_x^2+h_y^2} \;\; 0 \;\; h_z\,]$
$J^T_{g,b}({}^S_E\hat{q}_{est,t-1}, {}^E\hat{b}_t) f_{g,b}({}^S_E\hat{q}_{est,t-1}, {}^S\hat{a}, {}^E\hat{b}_t, {}^S\hat{m})$
$\int .dt$
$2\,{}^S_E\hat{q}^*_{est,t-1} \otimes {}^S_E\dot{\hat{q}}_{\epsilon,t}$
$\frac{\nabla f}{\|\nabla f\|}$
ζ
β
$\frac{1}{2}\,{}^S_E\hat{q}_{est,t-1} \otimes {}^S\omega_{c,t}$
${}^S_E\dot{q}_{est,t}$
$\frac{q}{\|q\|}$
z^{-1}
${}^S_E\hat{q}_{est,t}$

Here, the block diagram represents the complete orientation filter

We will try to implement the following code (https://github.com/bjohnsonfl/Madgwick_Filter/blob/master/madgwickFilter.c)

Mahony Filter:



IMU
${}^I\mathbf{a}_{t+1}$
${}^I\omega_{t+1}$
$\mathbf{v}\left({}^I_W\hat{\mathbf{q}}_{est,t}\right) = \begin{bmatrix} 2(q_2q_4 - q_1q_3) \\ 2(q_1q_2 + q_3q_4) \\ (q_1^2 - q_2^2 - q_3^2 + q_4^2) \end{bmatrix}$
$\mathbf{e}_{t+1} = {}^I\hat{\mathbf{a}}_{t+1} \times \mathbf{v}\left({}^I_W\hat{\mathbf{q}}_{est,t}\right)$
$\mathbf{e}_{i,t+1} = \mathbf{e}_{i,t} + \mathbf{e}_{t+1}\Delta t$
Δt
${}^I\omega_{t+1} = {}^I\omega_{t+1} + \mathbf{K}_p\mathbf{e}_{t+1} + \mathbf{K}_i\mathbf{e}_{i,t+1}$
$\frac{1}{2}{}^I_W\hat{\mathbf{q}}_{est,t} \otimes [0, {}^I\omega_{t+1}]^T$
${}^I_W\dot{\mathbf{q}}_{est,t+1}$
$\int$
${}^I_W\mathbf{q}_{est,t+1}$
Δt

Obtain gyro and acc measurements from the sensor. Let ${}^I\omega_t$ and ${}^I\mathbf{a}_t$ denote the gyro and acc measurements respectively. Also, ${}^I\hat{\mathbf{a}}_t$ denotes the normalized acc measurements.

https://nitinjsanket.github.io/tutorials/attitudeest/mahony

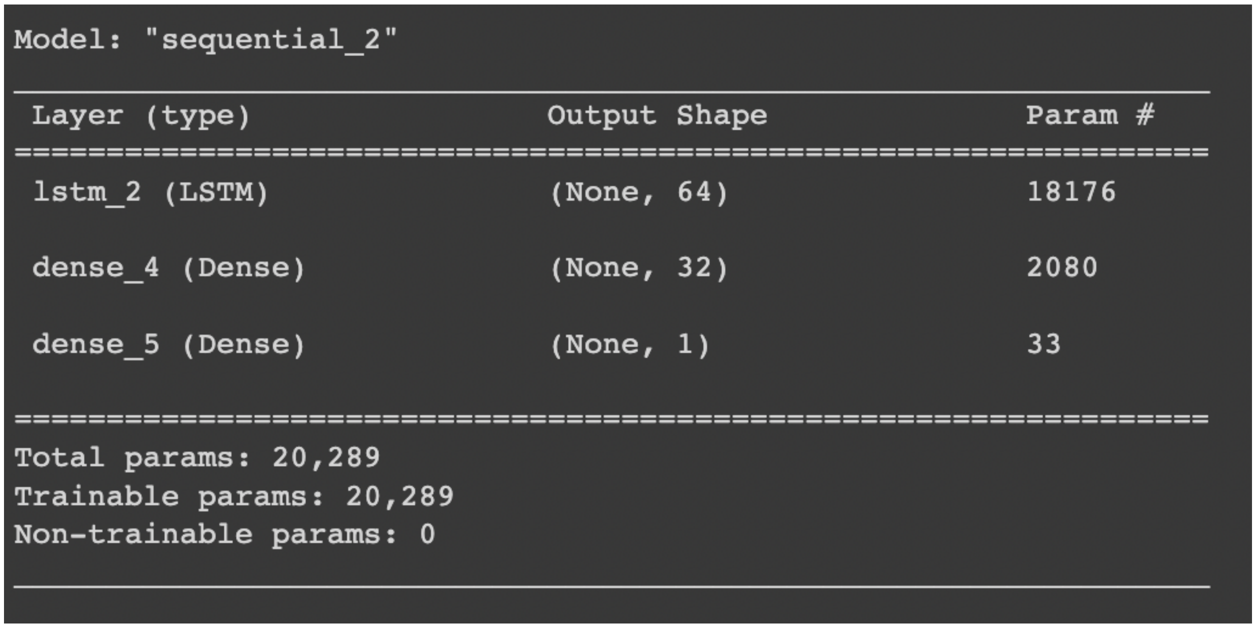
The Mahony filter is a type of complementary filter used to estimate the orientation of an object and is known for its high accuracy and fast processing times.

We will try to implement the following code (https://github.com/gaochq/IMU_Attitude_Estimator/blob/master/src/Mahony_Attitude.cpp)

Deep learning

When a large range of different dynamic and static rotational and translational motions is considered, the attainable accuracy is limited by the need for situation-dependent adjustment of accelerometer and gyroscope fusion weights. We investigate to what extent these limitations can be overcome by means of artificial neural networks and how much domainspecific optimization of the neural network model is required to outperform the conventional filter solution.

We have made an initial LSTM to test results. Although not greate, we can see a clear pattern in the predictions

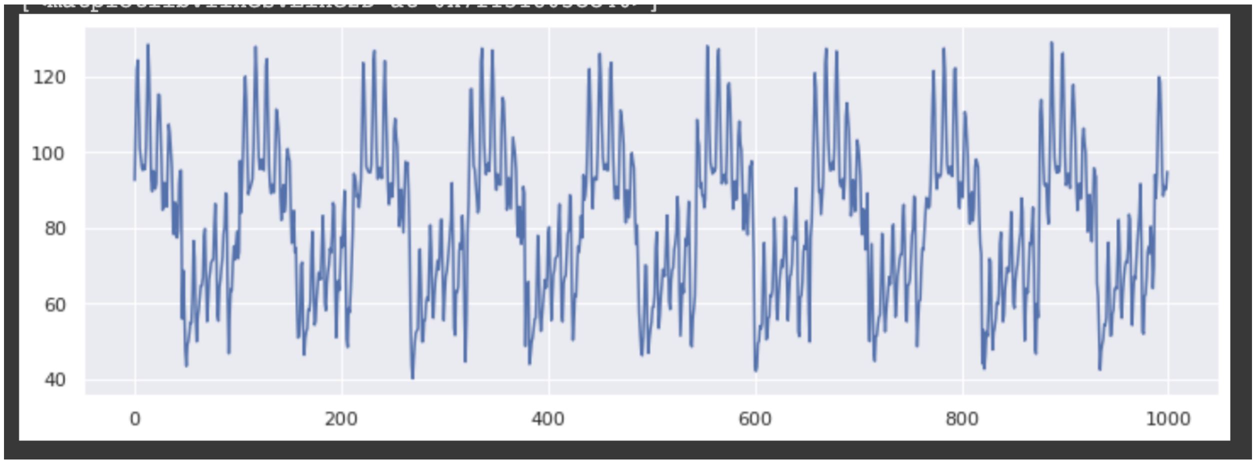


```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 64)                18176

 dense_4 (Dense)             (None, 32)                2080

 dense_5 (Dense)             (None, 1)                 33

=================================================================
Total params: 20,289
Trainable params: 20,289
Non-trainable params: 0
_________________________________________________________________
```



We will keep improving our models and deploy the ones from the following papers.

(https://arxiv.org/pdf/2005.06897.pdf)

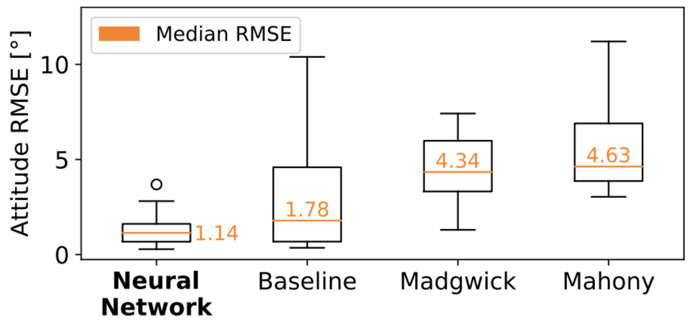
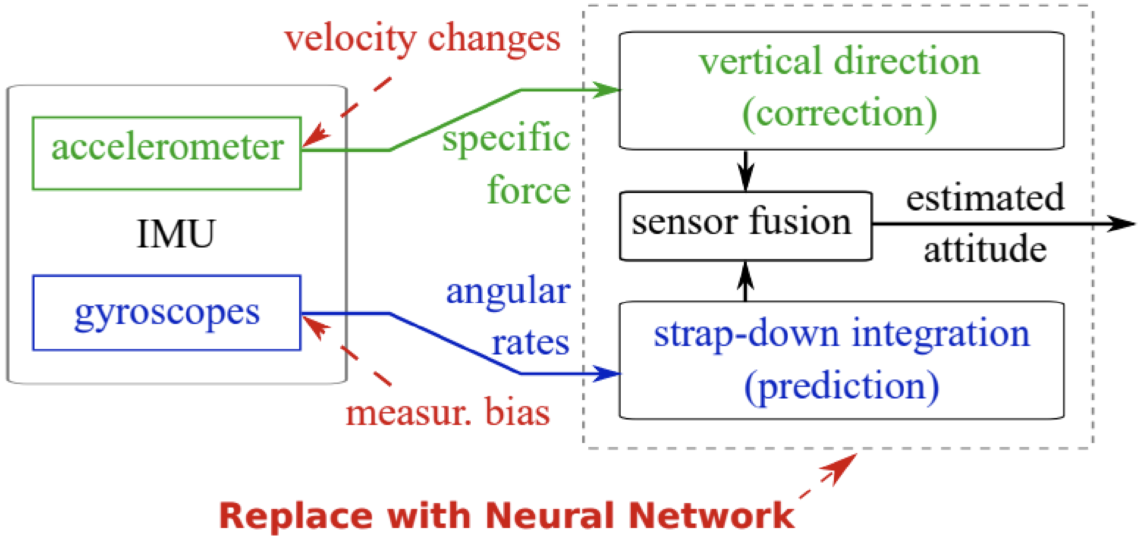
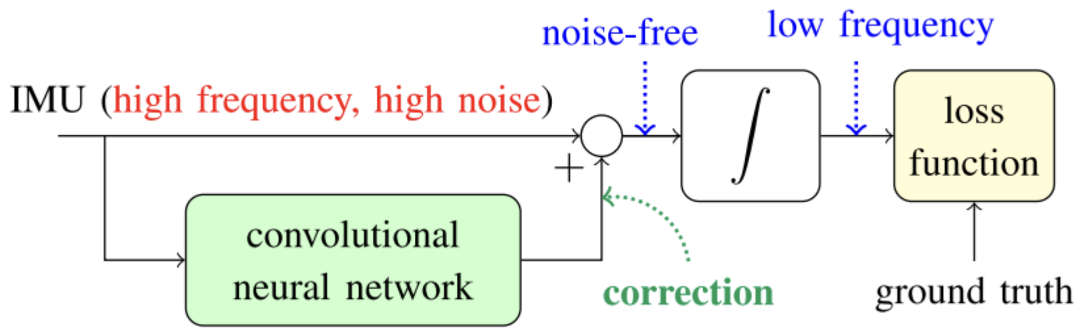
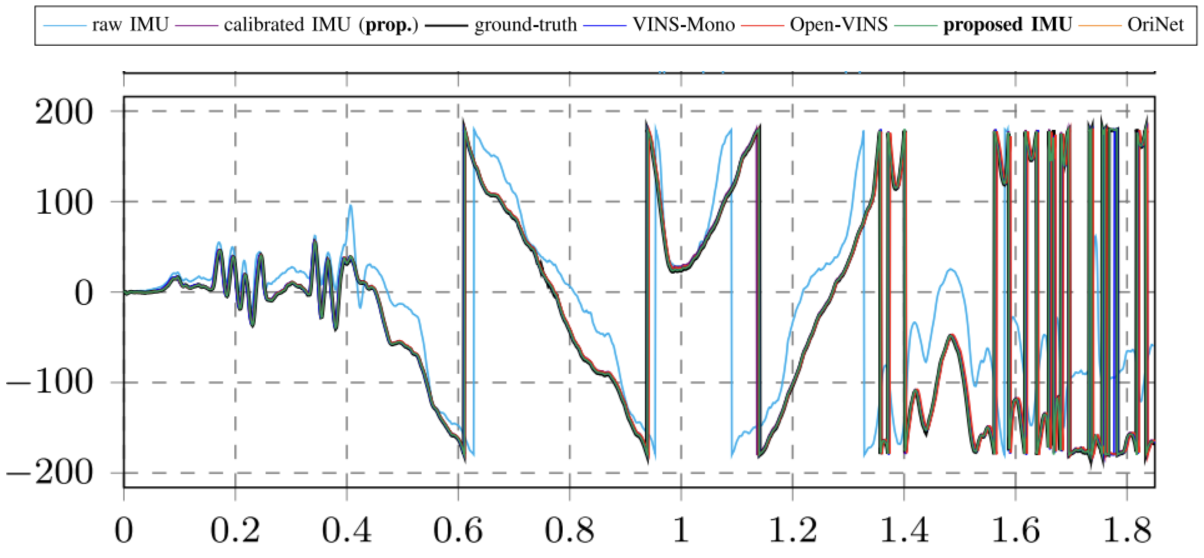


Fig. 8. RMSE comparison between the best neural network, the baseline filter [4], and two open-source available filter algorithms [31]. Across all types of motions, the proposed neural network achieves clearly smaller median and variance than the conventional filters. Details are presented in Fig. 9.



We can see based on results from papers that using a RNN or CNN minimizes the RMSE and provides promising results.
We will begin by implementing the following code (https://github.com/mbrossar/denoise-imu-gyro) for Denoising IMU Gyroscopes for Open-Loop Attitude Estimation. (https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9119813)

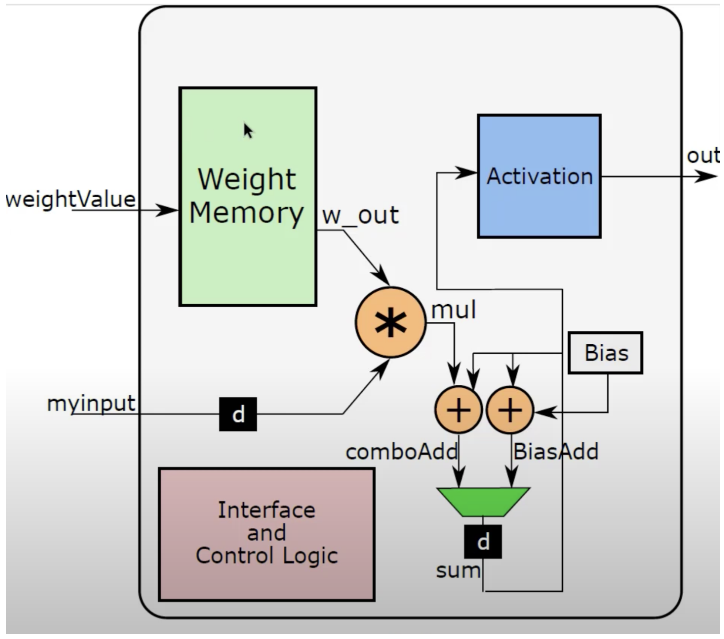

noise-free
low frequency
IMU (high frequency, high noise)
+
∫
loss function
convolutional neural network
correction
ground truth



This paper has promising results and we will compare the RMSE with that obtained by the standard algorithms. If this approach is significantly better, we will move onto accelerating this using hardware optimizations.

Neural Networks on FPGA:

The CNN is costly and will reduce the frequency of output. Therefore our reach goal is to deploy the CNN on an FPGA.



(https://www.youtube.com/watch?v=a2wOjxRf_xg&list=PLJePd8QU_LYKZwJnByZ8FHDg5l1rXtcIq&index=2)

Requirement	Verification
● The Algorithm used must be able to minimize RMSE on the dataset	● As we already know that standard models can perform with an RMSE of <6 degrees within 25 mins, the deep learning model needs to outperform these algorithms ● The Output of the model should be easy to understand and be subjected to unseen test data taken from other trials
● The Algorithm used should output the 3D rotation of the IMU	● The Algorithm will provide us with quaternions values that we need to use to display a 3D model on a screen ● The IMU can display the Euler angles on the LCD or OLED screen on the PCB.
● The Algorithm used can minimize the error over the data	● The calculated Mean absolute percentage error should be less than 5% over 20 mins of data collection

2.5 Subsystem 4: Power Supply

The Power Subsystem will serve as the primary source of energy for the various components in our design, including the IMU, microcontroller, AI on chip, and any hardware filters. This subsystem will perform the conversion of AC power from the wall outlet to DC power and regulate the DC power to meet the varying voltage requirements of the different components. For the power supply in the initial deliverable model we will use the 12V power supply that comes with the Nvidia Jetson developer kit to regulate the power that comes out of the wall socket and provide it to the Jetson.

For the initial deliverable model, to power the IMU and the control unit in the initial deliverable model we will use a USB cable to connect the PCB to the Jetson. The USB cable will not only transmit data between the microcontroller and the Jetson, but it will also supply the PCB with a 5v input voltage. We will then use a 3.3V voltage regulator to convert that voltage down to 3.3V which is what the ESP32 runs on. We can then use that reduced 3.3 voltage to power the IMU and OLED display. A USB 2.0 port can supply 500mA of current at 5V. This is enough to power everything on the PCB, including the ESP32, the IMU, and the OLED display. In standard mode the ESP32 consumes about 260mA of current which leaves plenty for the other components. The IMU does not consume more than 5mA of current and the OLED display does not draw more than 50mA of current when the display is fully illuminated.

For the reach model of our project we will no longer rely on the USB cable to deliver power to the control unit and IMU subsystems, instead we will use a proprietary voltage regulator such as the LM1117 connected to a DC power adapter such as the Belker 12W Universal AC/DC Power Adapter. The 12W power adapter should be able to supply more than enough energy to the PCB as the ESP32 along with the OLED display board and IMU does not consume more than 1.5 watts of power. We will have to choose a suitable FPGA for the reach model that does not require more than 10W of power as we will need to use a bigger power adapter in that case.

Requirement	Verification
● The power supply must provide a stable output voltage of 3.3 volts to the PCB and a stable output voltage of 12 volts to the NVIDIA Jetson.	● Measure the output voltage of the PCB power supply using a multimeter and verify that the value is 3.3 volts with a tolerance range of +/- 0.1 volts. ● Measure the output voltage of the Jetson power supply using a multimeter and verify that the value is 12 volts with a tolerance range of +/- 0.5 volts ● Place the PCB under varying load conditions and measure the output voltage to ensure it remains stable within the tolerance range.
● The power supply must be able to handle a maximum input voltage of 24	● Increase the input voltage of the power supply in small increments while measuring the output voltage until

volts with a tolerance range of +/- 0.5 volts.	the maximum voltage is reached, and record the output voltage value. ● Verify that the power supply can handle the maximum input voltage without damaging any components or causing any safety hazards.
● The power supply must have overvoltage protection to prevent any voltage spikes from damaging the components.	● Introduce a voltage spike to the input voltage of the power supply and measure the output voltage to ensure it remains within the tolerance range. ● Verify that the overvoltage protection activates and prevents any voltage spikes from passing through to the components.
● The power supply must have short-circuit protection to prevent any damage to the components in case of a short circuit.	● Introduce a short circuit to the output of the power supply and verify that it shuts down and prevents any current from flowing. ● Verify that the short-circuit protection activates and prevents any damage to the components.
● The power supply must have an efficiency of at least 85% to minimize power loss and heat generation.	● Measure the input and output power of the power supply and calculate the efficiency. ● Verify that the power supply meets the efficiency requirement under varying load conditions.

2.6 Tolerance Analysis

One aspect of our project that is critical to its success is whether the ESP32 can process the data as fast or faster than it receives the data from the IMU. This is important otherwise we can't get a real time pose estimation of the IMU.

Based on the known parameters of the IMU and the ESP32 , we can perform a tolerance analysis to determine whether the ESP32 can handle the data collection and processing requirements for the project. The IMU has a sample rate of 400 Hz and outputs data with 9 degrees of freedom. The data processing only requires applying constant integer offsets to the IMU data.

With a raw data output rate of 3.2 Mbps, the data transfer from the IMU to the ESP32 will not be a limiting factor, as the ESP32 has a maximum data transfer rate of 480 Mbps. We can estimate the computational requirements for processing the IMU data by considering the worst-case scenario of having to process all 400 samples per second. Each sample contains 9 values, each requiring one integer offset calculation. This results in a total of 3,600 integer calculations per second.

The ESP32 has a clock speed of 240 MHz, which means it can perform approximately 480,000 integer calculations per second. Therefore, the ESP32 is more than capable of handling the required processing for the project, even in the worst-case scenario. The maximum number of integer calculations required per second is well below the ESP32's capabilities, and the data transfer rate between the IMU and ESP32 is not a limiting factor.

In conclusion, the tolerance analysis shows that the ESP32 has ample processing power to handle the data collection and processing requirements of the project. Therefore, we can confidently state that this aspect of the project is feasible and can meet its requirements.

3. Cost and Schedule

3.1 Cost Analysis

Labor: 40$/hour x 2.5 x 8 hours per week x 7 weeks x 2 partners = $10,200
The total cost for parts as seen in the table below is $206.71 before shipping. 5% shipping cost adds another $10.34 and another 10% sales tax adds another $20.67. This adds up for a total of $237.71

Adding up labor and parts the total cost for this project comes out to be $10,437.71

Part #/Description	Distributer	Quantity	Total Cost
MPU-6050 IMU	Amazon	3	$10
MPU-9250 IMU	Amazon	1	$15
ESP32 Microcontroller	Digikey	1	$5
128x64 OLED Display	Amazon	1	$7
SDSDXXU-064G-GN4IN 64GB SD card	Amazon	1	$15
Nvidia Jetson Developer Board	Nvidia	1	$150
AA3528QBS/D 465nm LED	Kingbright	6	$2.76
1k Ohm Resistor	Yageo	6	$0.60
1825910-6 push button	TE Connectivity	3	$0.60
612-EG1201A Switch	Mouser Electronics	1	$0.75
MCP2200T-I/MQ USB 2.0 to UART	Mouser Electronics	1	$2.84

Protocol Converter			

3.2 Schedule

Dates	**Tasks**
2/19-2/25 Design Document 2/23 Team Contract 2/24	Lukas: Finish up design document and team contract and start designing PCB Chirag: Finish up design document and team contract and Finish implementing the standard models on test IMU data (contact lab)
2/26 - 3/4 Design Review	Lukas: Finalize first draft of PCB design and start testing ESP32 and IMU's Chirag: Finish one of the models from the papers that we have mentioned (Start with CNN) and compare performance with standard model results
3/5 - 3/11 First Round PCB 3/7 Teamwork Evaluation 3/8	Lukas: Start constructing mock up of pcb using a breadboard to ensure that all the components work together and are able to interface as expected Chirag: Finish deep learning models from the papers and start using the Jetson for displaying the final 3d estimate using all algorithms and assist with pcb design
3/12 - 3/18	SPRING BREAK
3/19 - 3/25	Lukas: Start testing pcb and order more parts if needed Chirag: Begin testing hardware acceleration for the Models and assist in testing pcb
3/26 - 4/1 Second Round PCB 3/28 Progress Report 3/29	Lukas: Start designing second round pcb with reach model in mind Chirag: Assist in testing second round pcb and move forward depending on results from the Deep learning models and acceleration hardware
4/2 - 4/8	Lukas: Finalize second round pcb design Chirag: Assist in finalizing second round pcb
4/9 - 4/15	Lukas: Finalize testing the for bugs and ensuring the hardware works Chirag: Finalize testing the for bugs and ensuring the hardware works
4/16 - 4/22 Team Contract 4/16 MOCK DEMO WEEK	Lukas: Finalize construction and assembly of final design of project Chirag: Finalize construction and assembly of final design of project

4/23 - 4/29 FINAL DEMO WEEK	Lukas: Prepare for final demo and work on final report Chirag: Prepare for final demo and work on final report
4/30 - 5/6 FINAL PRESENTATION	Lukas: Prepare for final presentation Chirag: Prepare for final presentation

4. Discussion of Ethics and Safety

4.1 Ethics and Safety Issues

In accordance with the IEEE Code of Ethics, it is of utmost importance to maintain originality and integrity in the project ideas and research process. Any sources used during the research must be properly cited and credited to avoid plagiarism (IEEE Code of Ethics II.5). Our project is aligned with the ongoing efforts in improving the accuracy of IMU sensors, and while referencing relevant research papers, all sources used will be properly cited and credited. Our project aims to differentiate itself from existing technologies by utilizing unique methods of implementation. Furthermore, the IEEE Code of Ethics I.5 requires that all claims and estimates be honest and realistic. In the context of our project, we strive to enhance the precision of IMU data output to the best of our abilities, which includes verifying the reliability of raw data from various IMU sensors.

In terms of safety, our team is committed to following the laboratory safety regulations set by the Division of Research Safety in the Office of the Vice Chancellor for Research and Innovation (ECE 445 p.3). To minimize any potential risks, our team will work in pairs during laboratory sessions, promptly report any broken equipment, maintain cleanliness after each session, and avoid consuming food within the lab.The main concern of safety within out project is that of a short circuit within our PCB or power supply that cause elements of the electrical circuit to overheat and burn the users of the project. To mitigate this risk we will exercise caution to avoid skin contact with electrical circuits and separate electrical systems with high voltage requirements from human interaction. We will also thoroughly test our circuits against varying voltages and currents to make sure that a power surge will not cause any potential damage or safety hazards within our project.

5. Citations and References

[1] X-IO Technologies. “NGIMU: The Complete Sensor Fusion Solution.” X-IO Technologies, www.x-io.co.uk/ngimu/.

[2] X-IO Technologies. “NGIMU: Sensor Fusion with IMU & AHRS.” YouTube, uploaded by X-IO Technologies, 9 Apr. 2018, www.youtube.com/watch?v=a2wOjxRf_xg&list=PLJePd8QU_LYKZwJnByZ8FHDg5l1rXtcIq&index=2.

[3] "IEEE Code of Ethics." IEEE, www.ieee.org/about/corporate/governance/p7-8.html.

[4] Association for Computing Machinery. “ACM Code of Ethics.” Association for Computing Machinery, www.acm.org/code-of-ethics.

[5] Kim, Hyunseok, et al. “Evaluating the Accuracy of Inertial Measurement Units: A Comparative Study on Real-World Data.” arXiv, Cornell University Library, 8 May 2020, arxiv.org/pdf/2005.06897.pdf.

[6] Brossard, Marc. “Denoise-Imu-Gyro.” GitHub, github.com/mbrossar/denoise-imu-gyro.

[7] Adafruit Industries, LLC. “Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout - BNO055.” Adafruit Industries, LLC, www.adafruit.com/product/4502.

[8] HiLetgo. “HiLetgo MPU-6050 3 Axis Accelerometer Gyroscope Module 6 DOF IMU for Arduino.” Amazon, www.amazon.com/HiLetgo-MPU-6050-Accelerometer-Gyroscope-Converter/dp/B01DK83ZYQ?th=1.

[9] Hrisko, J. (2019). Accelerometer, Gyroscope, and Magnetometer Analysis with Raspberry Pi Part I: Basic Readings. Maker Portal. https://makersportal.com/blog/2019/11/11/raspberry-pi-python-accelerometer-gyroscope-magnetometer

[10] Shimmer Sensing. “Wireless Sensor Networks Videos.” Shimmer Sensing, shimmersensing.com/support/wireless-sensor-networks-videos/.

[11] Adafruit. “Magnetometer Calibration.” Adafruit Learning System, learn.adafruit.com/ahrs-for-adafruits-9-dof-10-dof-breakout/magnetometer-calibration.

[12] Hackster.io. “MPU9250 Datasheet.” TDK InvenSense, 2015, invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf.

[13] Christopher Barnatt. "Raspberry Pi Pico Tutorial #5: Pico and Accelerometers." YouTube, uploaded by ExplainingComputers, 2 Feb. 2021, www.youtube.com/watch?v=XCyRXMvVSCw.

[14] Christopher Barnatt. "Raspberry Pi Pico Tutorial #6: Pico and Gyroscopes." YouTube, uploaded by ExplainingComputers, 6 Feb. 2021, www.youtube.com/watch?v=mzwovYcozvI.

[15] IMU Filters https://nitinjsanket.github.io/tutorials/attitudeest/mahony