Communication & Controls in Iot

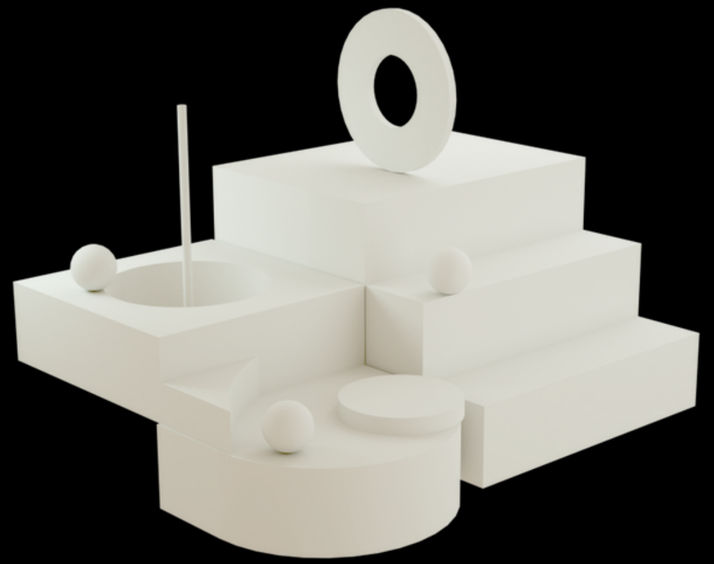
2-WHEELER SAFETY

DATA LOGGER

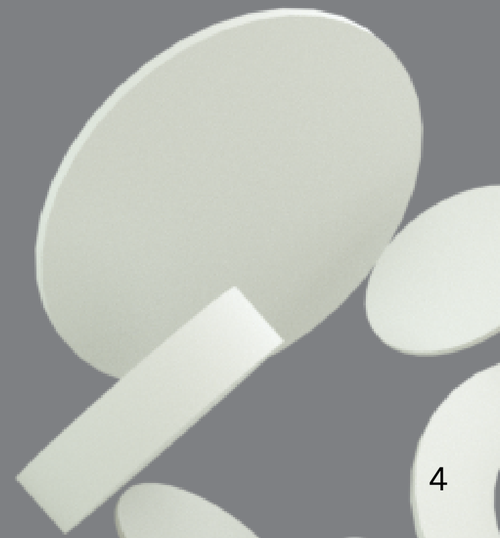
Acceleration, Location and Angular Acceleration

MOTIVATION and APPLICATIONS

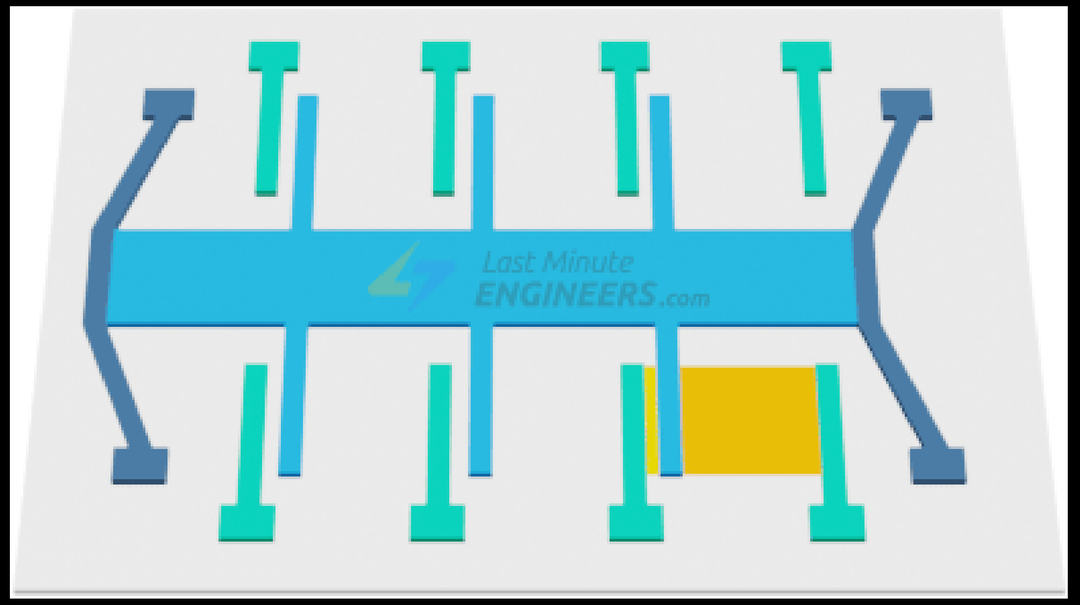
- Measuring the bumpiness of the road.
- Road quality data can be used by the government for road safety.
- To give the travellers data about which roads to avoid and which route is the safest.
- Analysis of accidents.
- Identify accident-prone areas
- Analysis of speed limits in different locations.



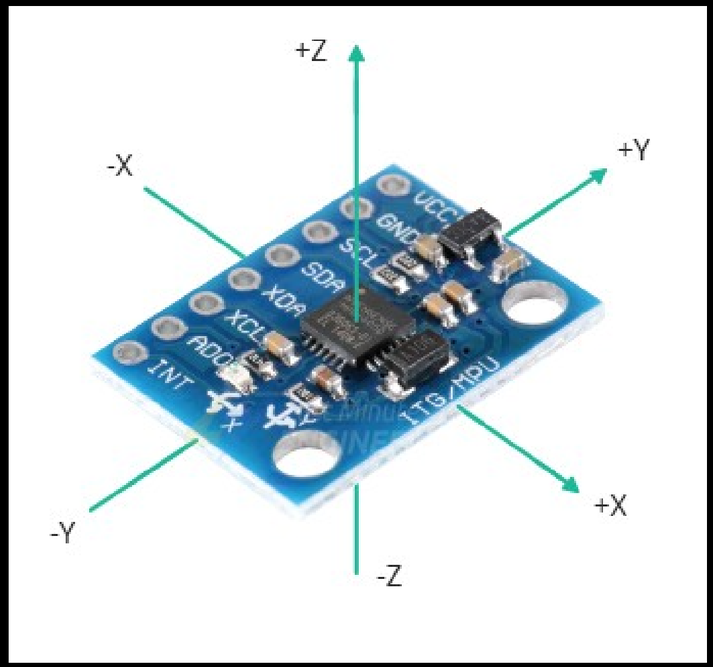
ACCELEROMETER



What is an Accelerometer?



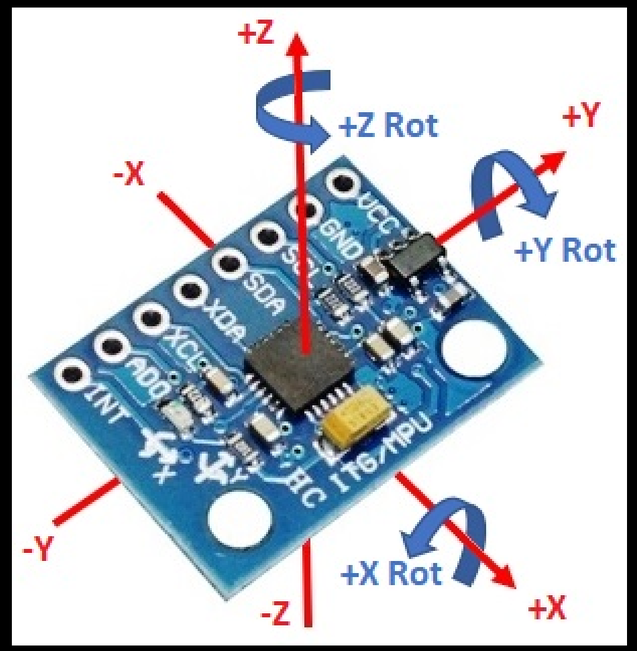
- It measures linear acceleration in all 3 directions
- A MEMS (Micro-Electro-Mechanical System) silicon wafer structure is used
- How does it measure??

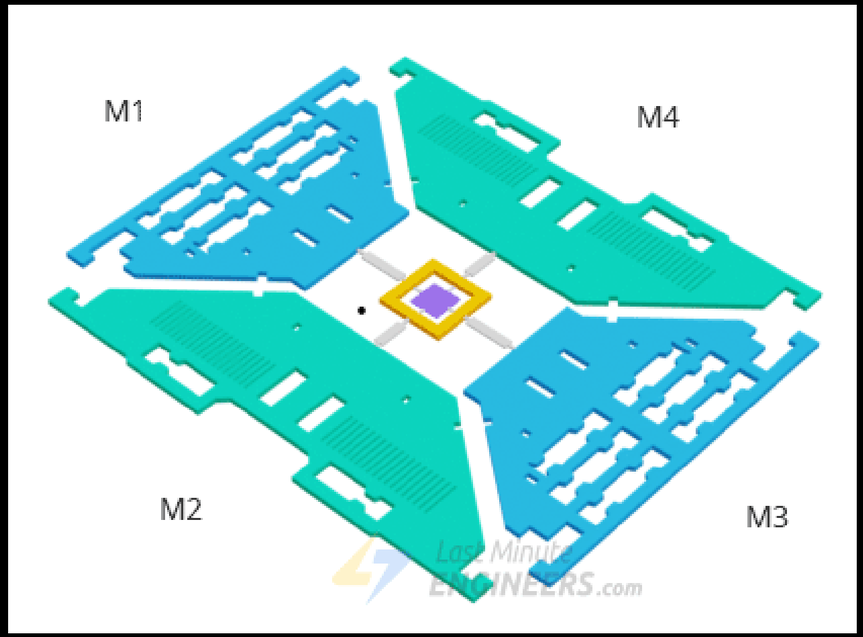


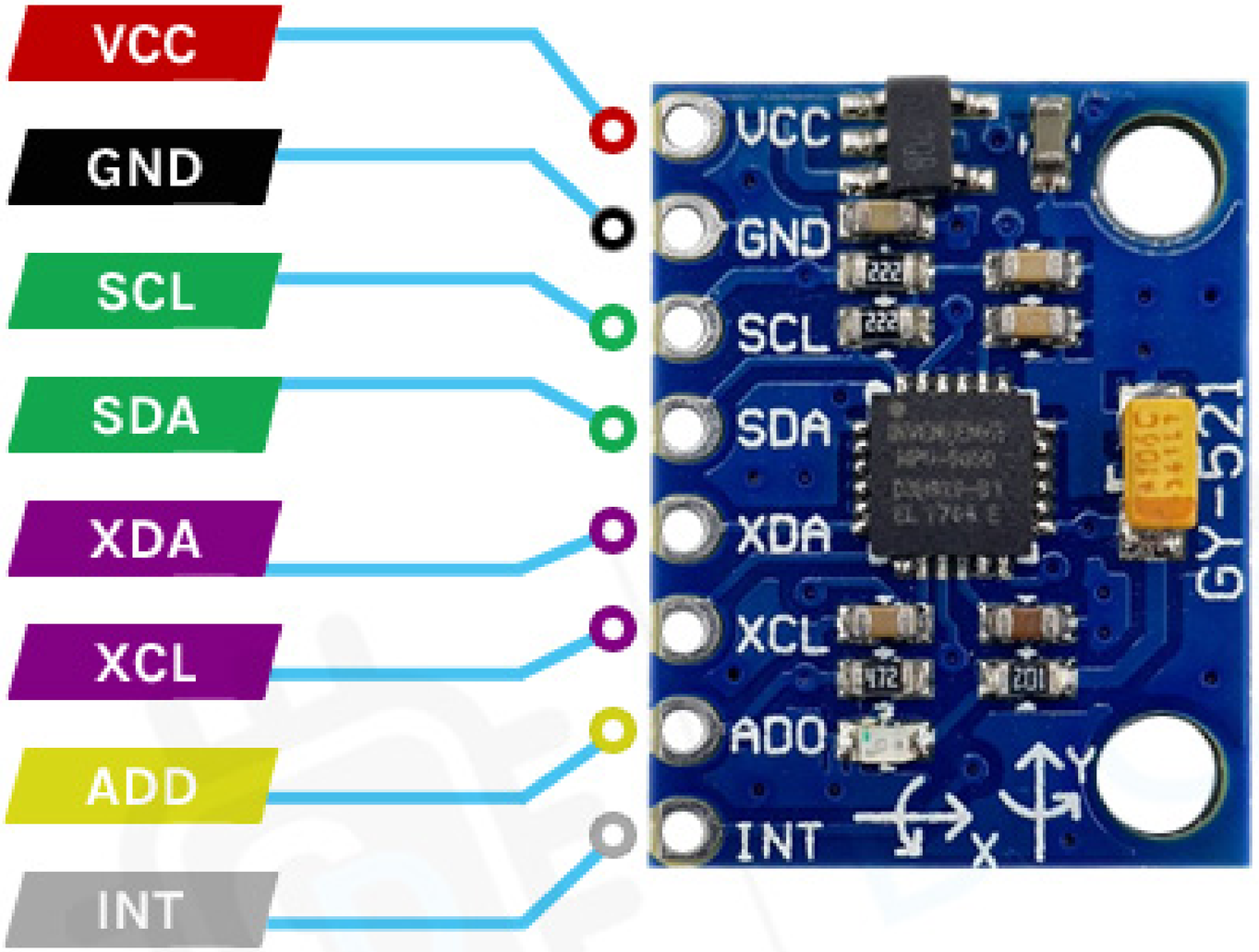
GYROSCOPE

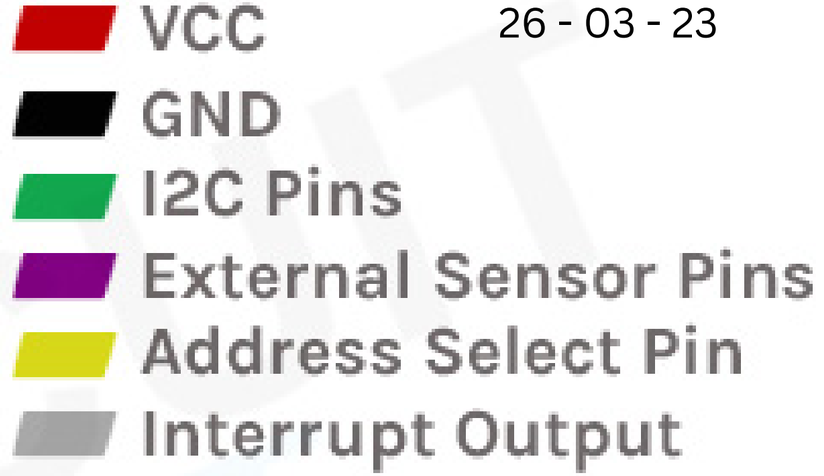
What is a Gyroscope?

- A sensor used to measure the orientation and angular velocity of an object.
- Measures in degree per second
- Installed in applications where the orientation of the object is difficult to sense by humans.

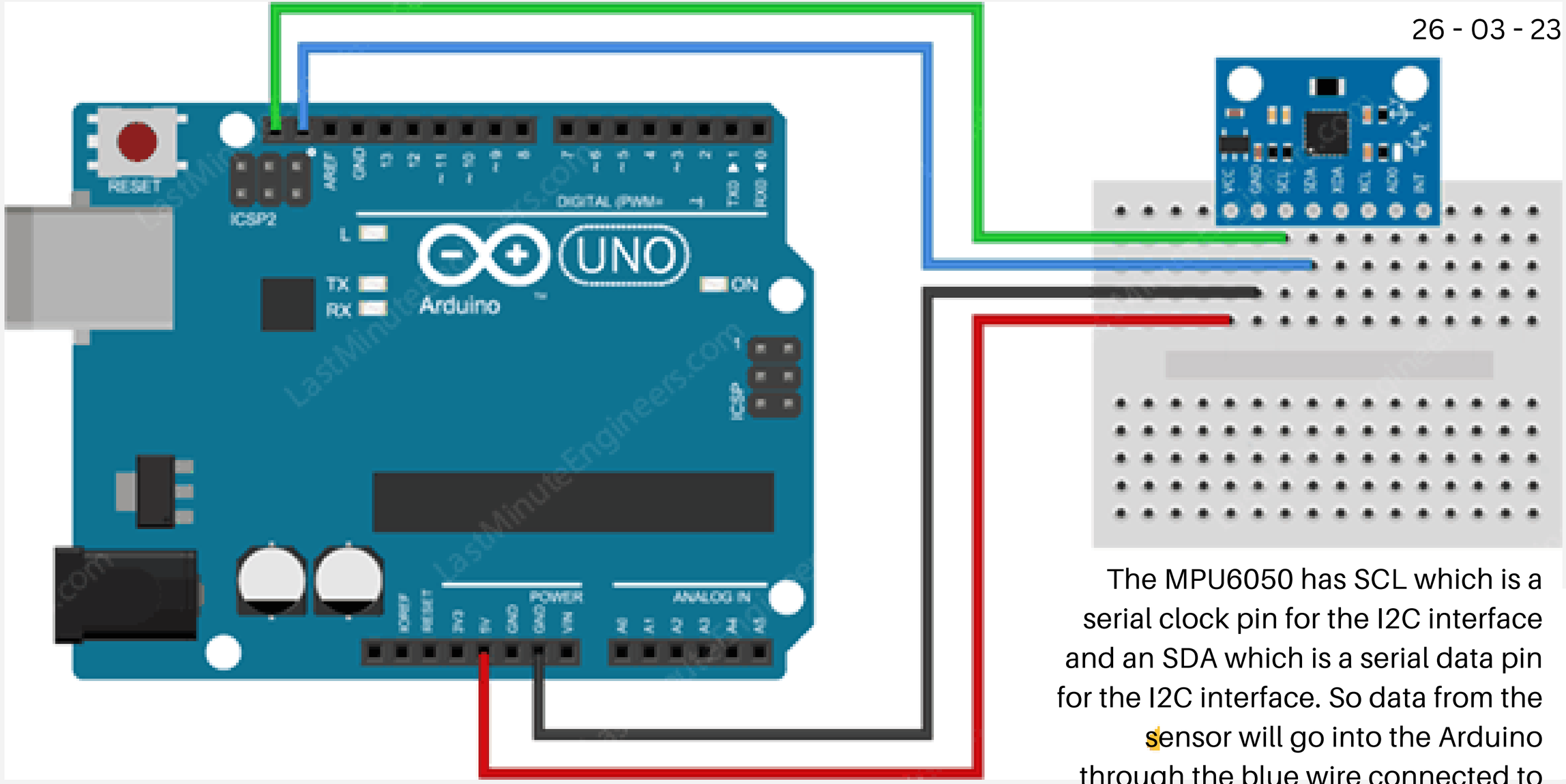








MPU6050 Module Pinout



The MPU6050 has SCL which is a serial clock pin for the I2C interface and an SDA which is a serial data pin for the I2C interface. So data from the sensor will go into the Arduino through the blue wire connected to the A4 pin of the Arduino.

Connecting MPU6050 to Arduino UNO

Measuring Acceleration

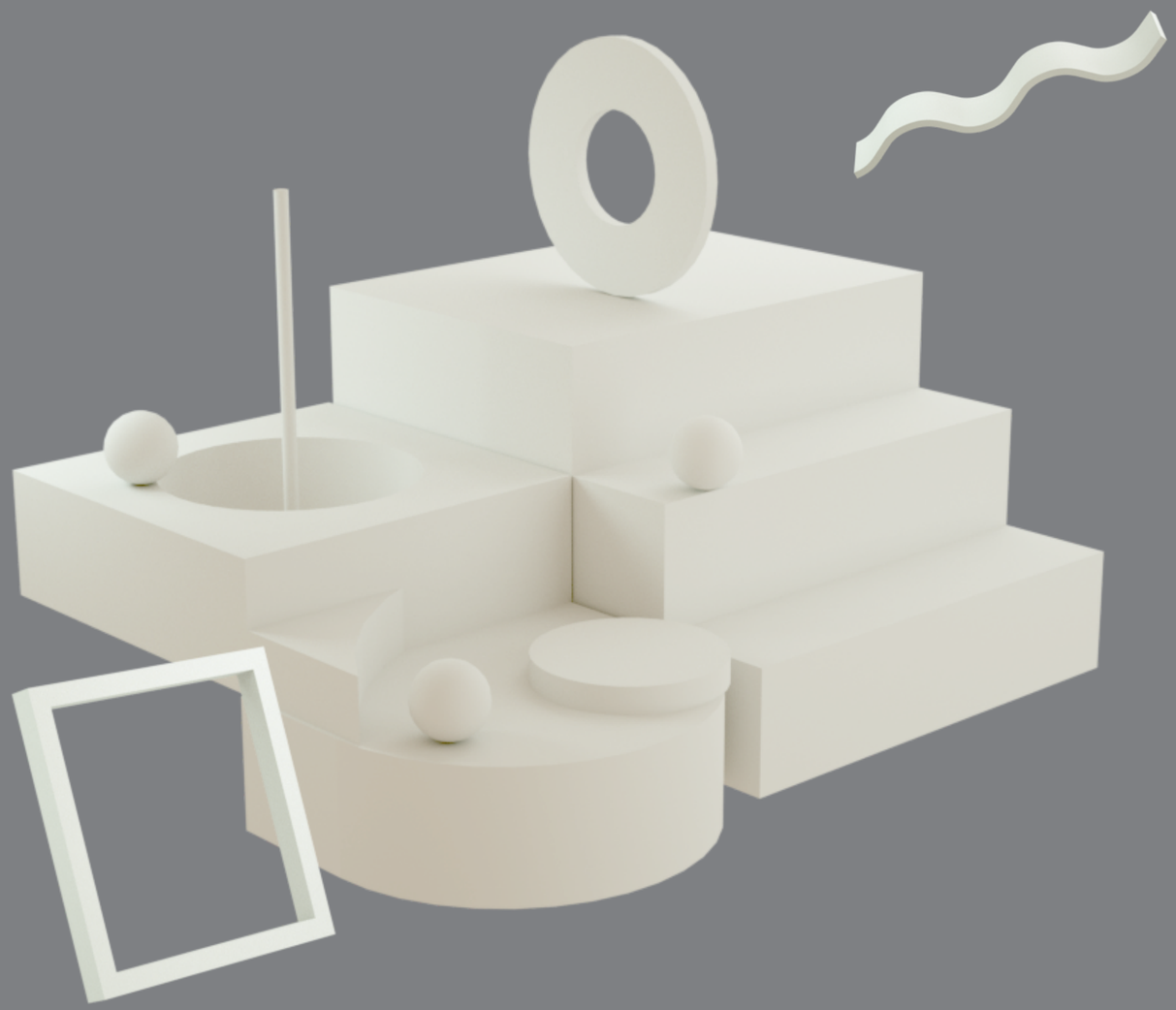
Using MPU6050 Module

- Can measure acceleration over four programmable full-scale ranges of ±2g, ±4g, ±8g, and ±16g.
- Equipped with three more 16-bit analog-to-digital converters that simultaneously sample the three axes of movement (along the X, Y, and Z axes).
- The Wire.h Library will be used in the Arduino code to get acceleration information inputs from the accelerometer and convert them into m/s^2

Measuring Rotation

Using MPU6050 Module

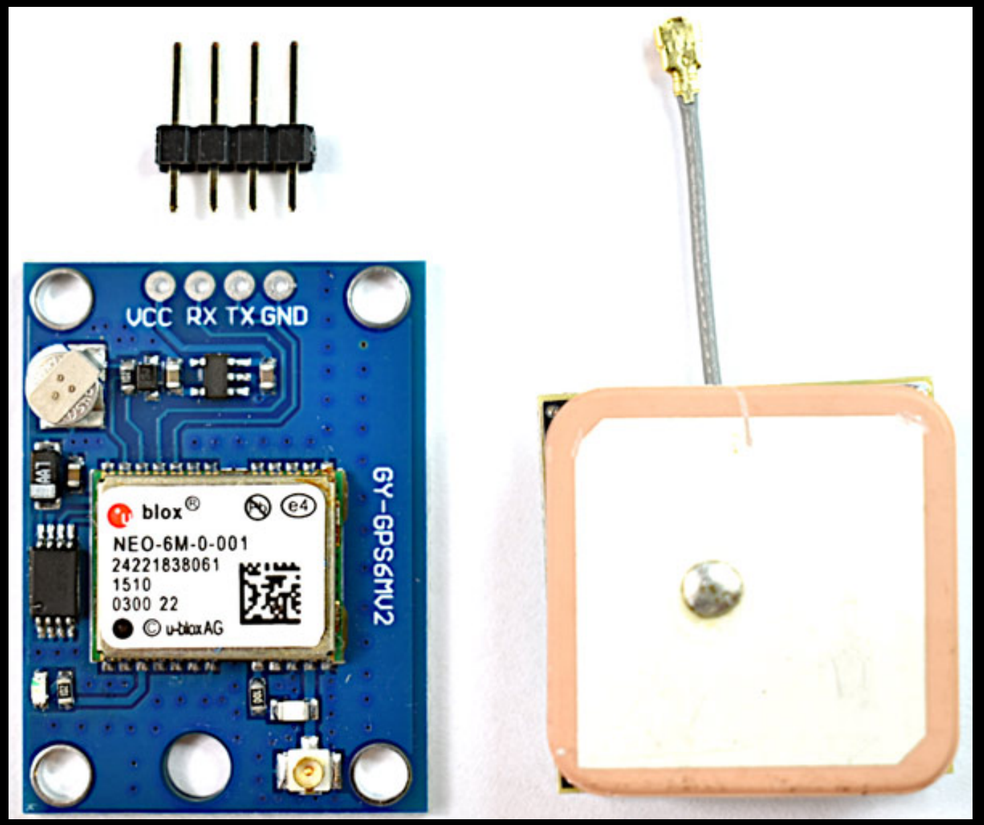
- Can measure angular rotation over four programmable full-scale ranges of ±250°/s, ±500°/s, ±1000°/s, and ±2000°/s.
- Equipped with three more 16-bit analog-to-digital converters that simultaneously sample the three axes of rotation (along the X, Y, and Z axes).
- The Wire.h Library will be used in the Arduino code to get rotation information inputs from the gyroscope and convert them into radians.



NEO 6M:
GPS module

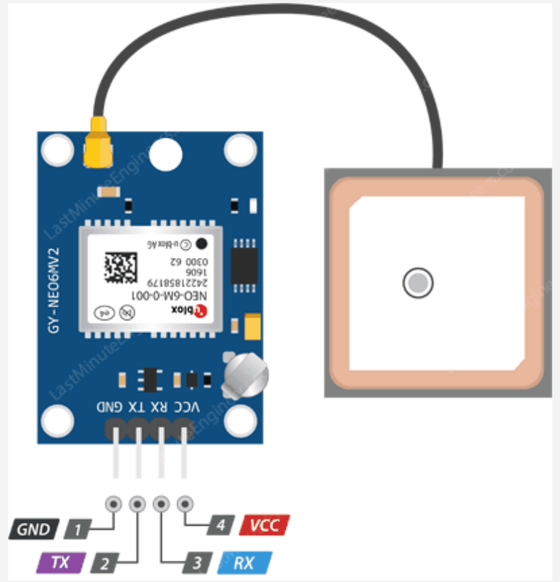
Basic Description:

- No blinking – it is searching for satellites.
- Blink every 1s – Position Fix is found (the module can see enough satellites).
- Acquires data from 22 sat's.
- UART connection.



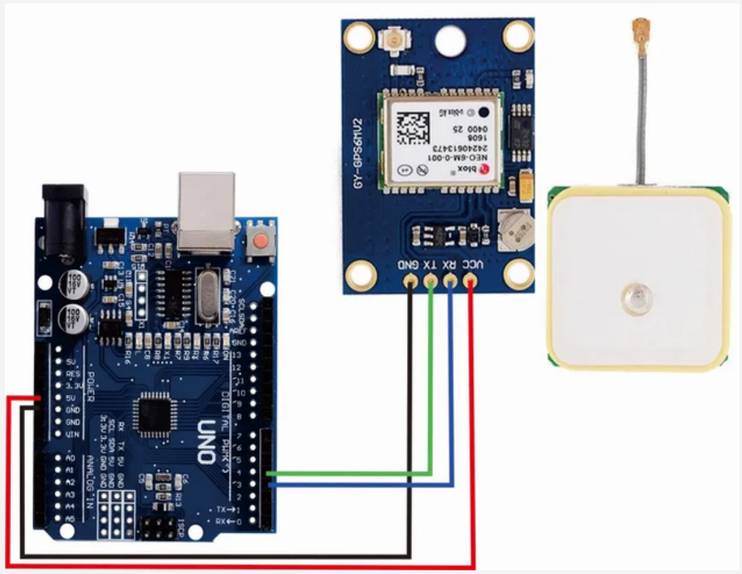
Hardware :

1. Arduino will provide 5V of input voltage, we will have to step it down to 3.3V.

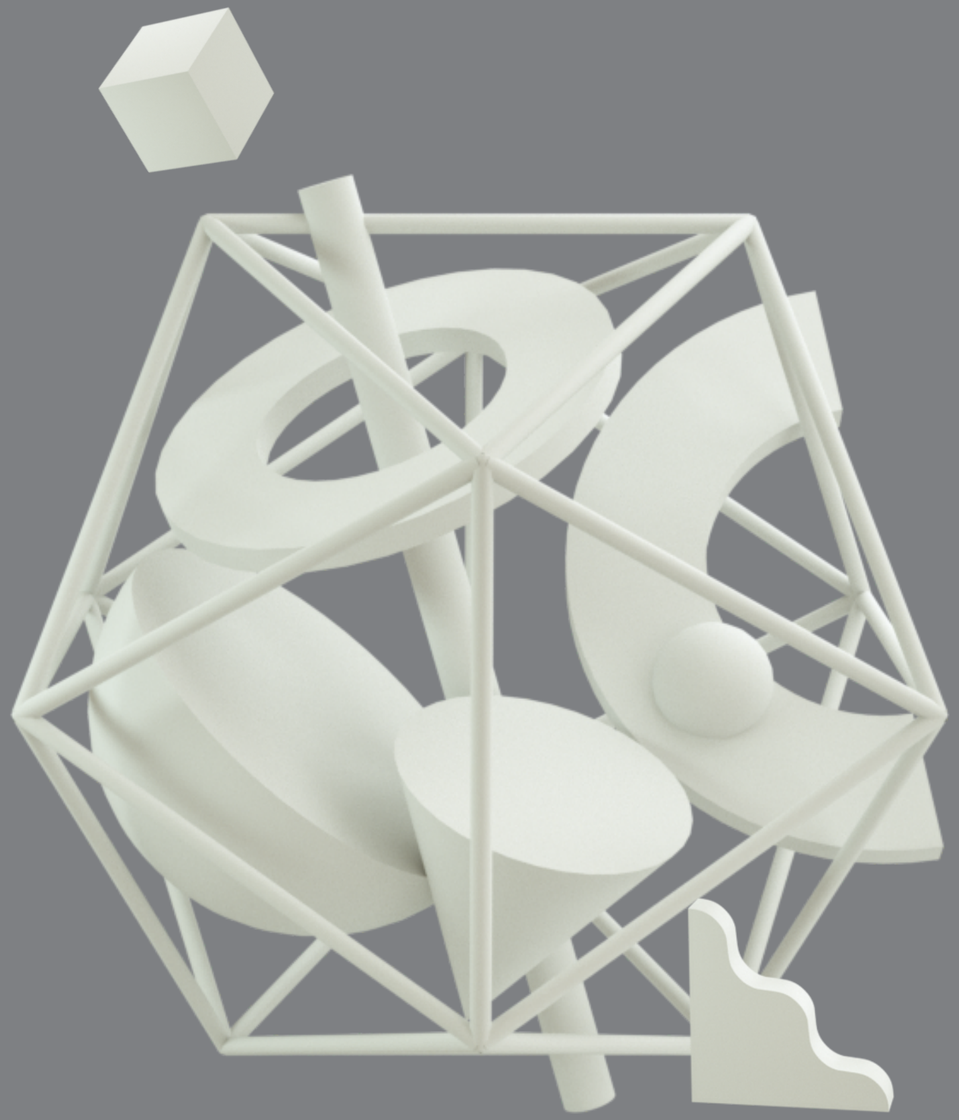


Software:

1. We used an Arduino library to parse the GPS data: **TinyGPSPlus library.**



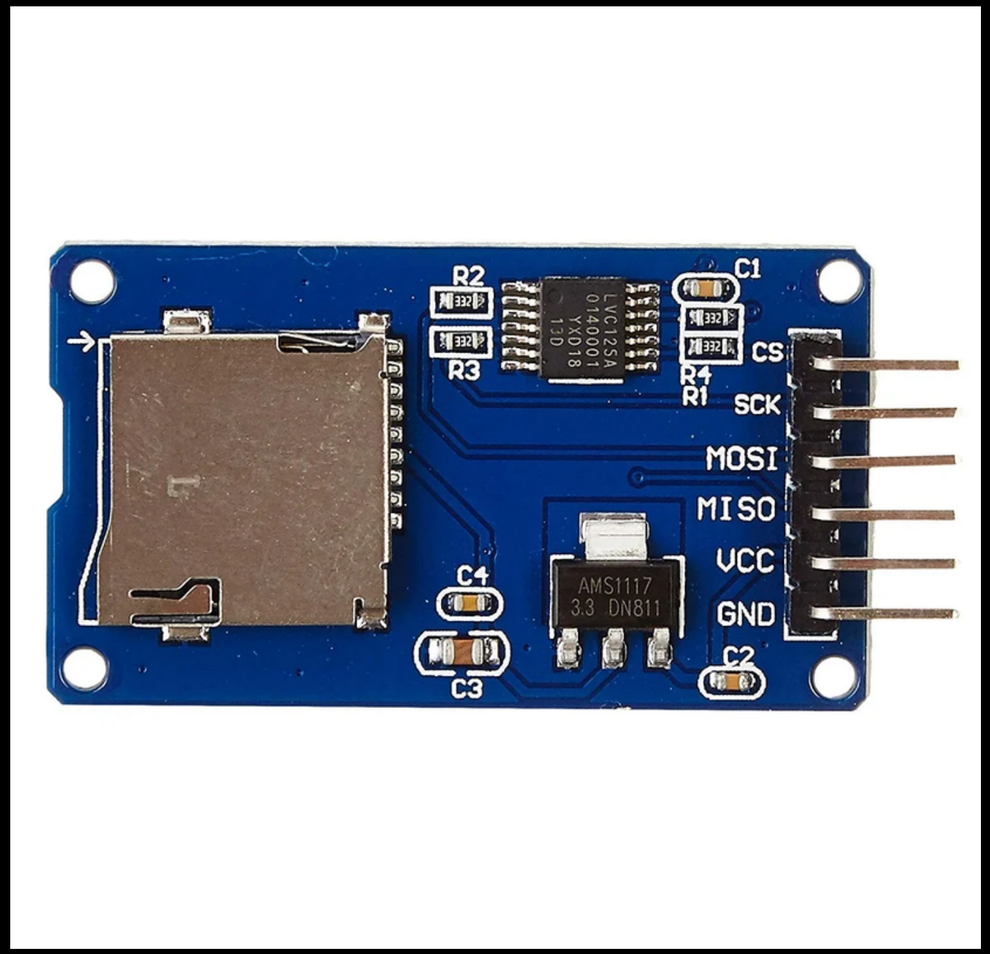
Date	Time	y_degree	y	x_degree	x
25/3/2023	11:17:36	17.445627	17°26'44N	78.348182	78°20'53E
25/3/2023	11:17:38	17.445627	17°26'44N	78.348182	78°20'53E
25/3/2023	11:17:40	17.445627	17°26'44N	78.348182	78°20'53E
25/3/2023	11:17:40	17.445627	17°26'44N	78.348182	78°20'53E
25/3/2023	11:17:44	17.445627	17°26'44N	78.348182	78°20'53E
25/3/2023	11:17:46	17.445627	17°26'44N	78.348182	78°20'53E
25/3/2023	11:17:50	17.448427	17°26'54N	78.345191	78°20'42E
25/3/2023	11:17:50	17.448427	17°26'54N	78.345191	78°20'42E
25/3/2023	11:17:50	17.448427	17°26'54N	78.345191	78°20'42E
25/3/2023	11:17:50	17.448427	17°26'54N	78.345191	78°20'42E



Storing Data on SD Card

Why SD Card?

- The SD card, or Secure Digital card, is a non-volatile solid-state flash memory.
- With an SD card, we can expand Arduino's permanent storage by gigabytes.



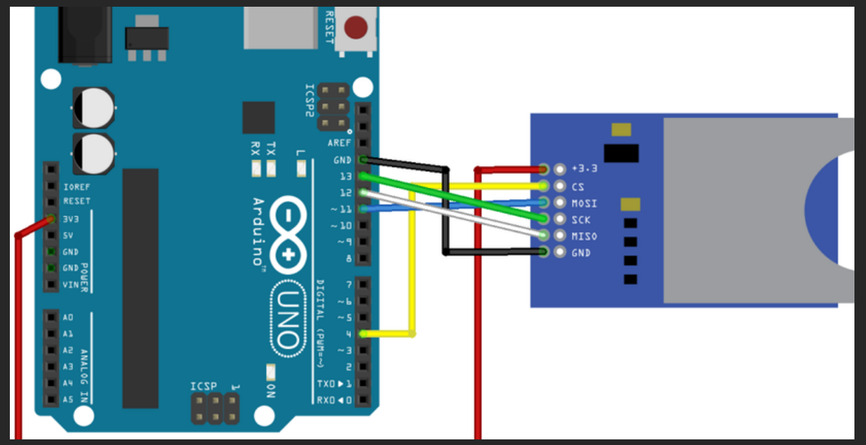
Using the SD card module with Arduino

Step 1

Format the card with a file system. The most common file systems for SD cards are FAT16 and FAT32.

Step 2

Testing the SD card:
Open the Arduino IDE software and go to File> Examples > SD > CardInfo.
Open the Serial Monitor and If everything is working properly, we can see a message on the serial monitor like the one shown.



Output Serial Monitor ×

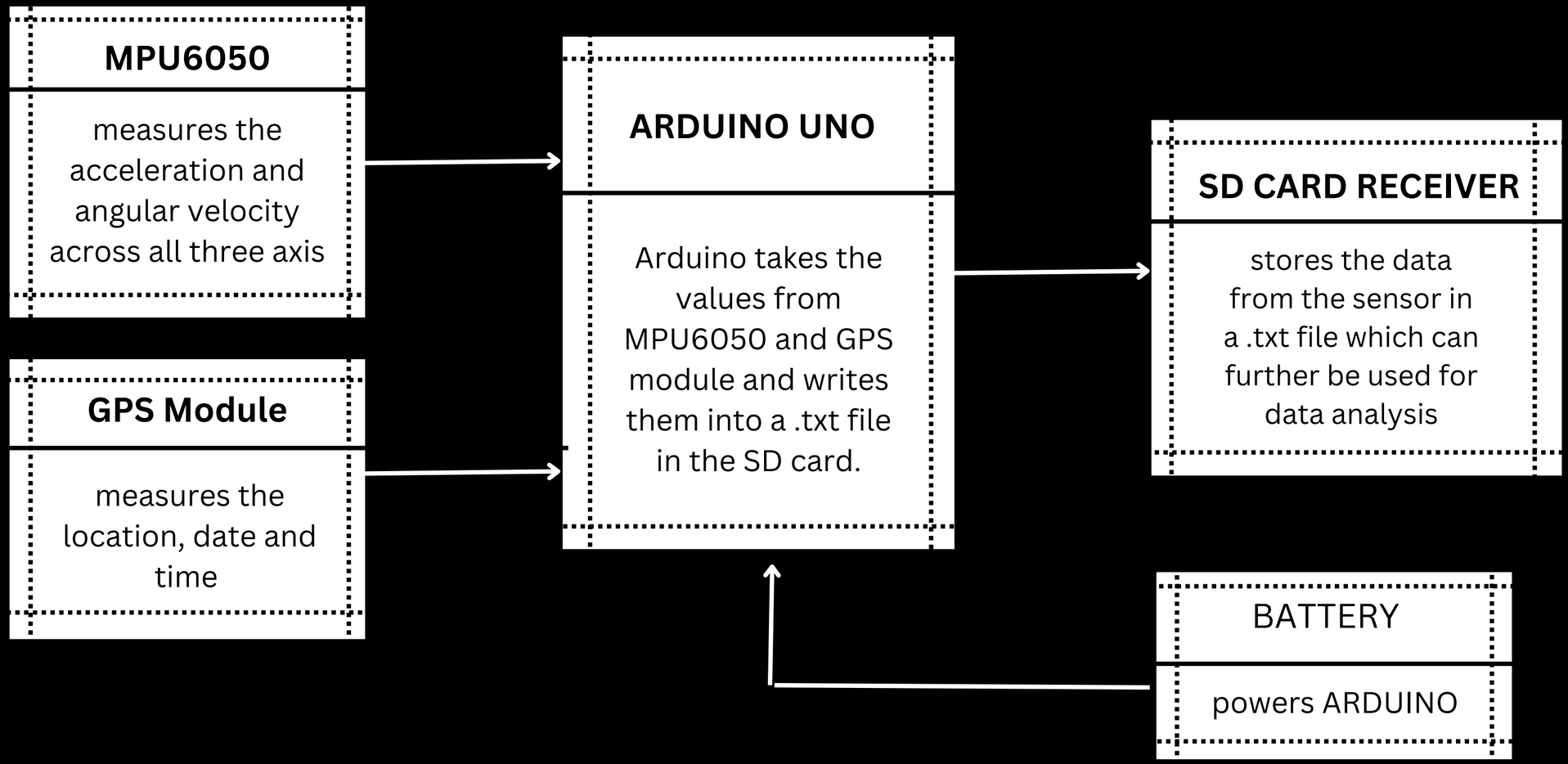
Message (Enter to send message to 'Arduino Uno' on 'COM5')

```
Initializing SD card...Wiring is correct and a card is present.

Card type:          SDHC
Clusters:           973952
Blocks x Cluster:   64
Total Blocks:       62332928

Volume type is:     FAT32
Volume size (Kb):   31166464
Volume size (Mb):   30436
Volume size (Gb):   29.72
```

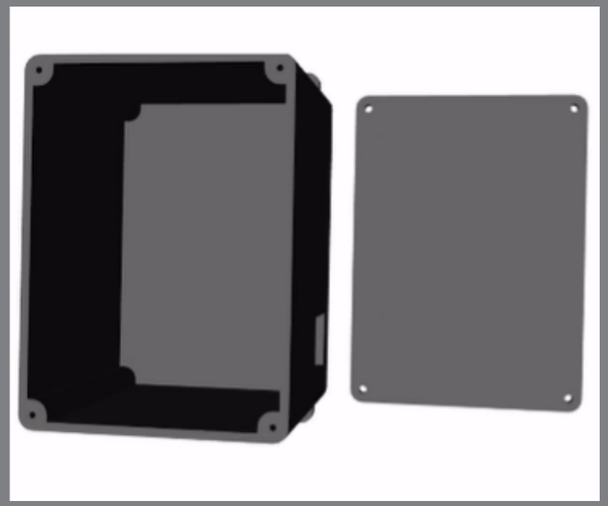
Read and Write to SD card

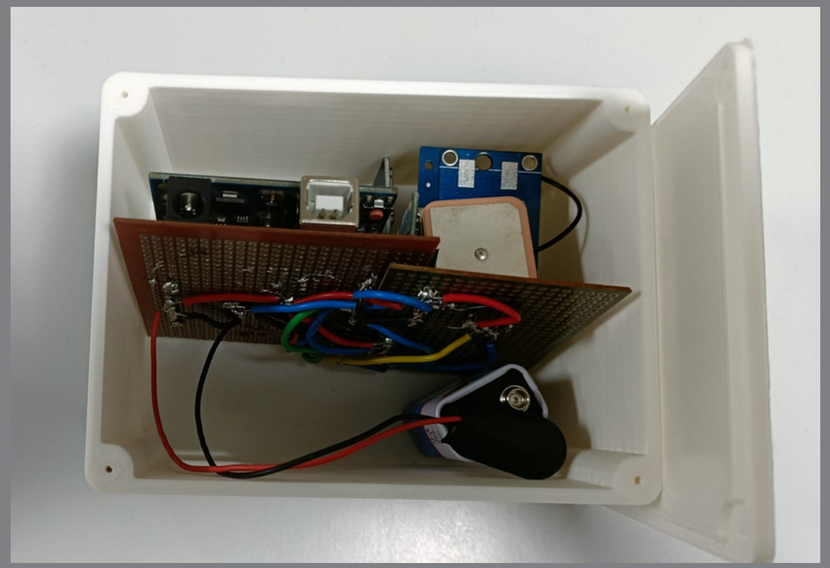
1. Include the necessary libraries (***SPI and SD***) for SD card reading and writing.
2. Initialize the SD card by calling the ***SD.begin()*** function in your setup() function.
3. Create a file on the SD card using the ***SD.open()*** function then specify the filename and the mode in which to open the file.
4. Write data to the file using the ***File.print()*** function.

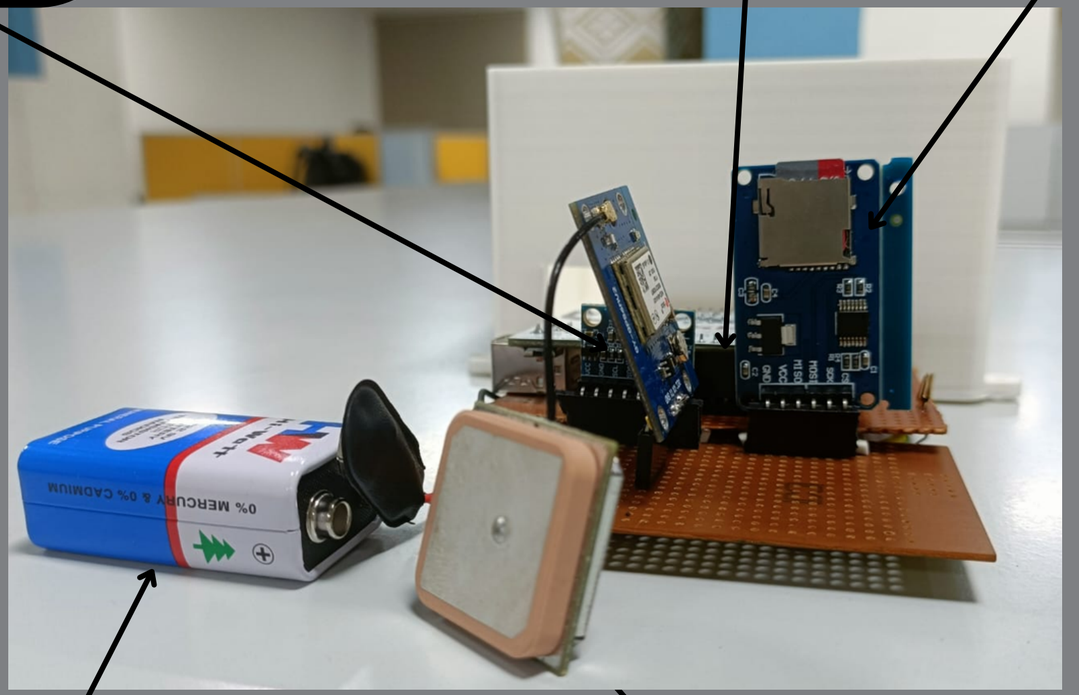
Block Diagram:



Hardware:







Code for MPU6050:-

```
// Library to include for MPU6050
#include <Wire.h>
```

```
const int MPU = 0x68; // MPU6050 I2C address
float AccX, AccY, AccZ, GyroX, GyroY, GyroZ;
float AccErrorX, AccErrorY, AccErrorZ, GyroErrorX, GyroErrorY, GyroErrorZ;
int c = 0;
```

```
// setting up MPU6050
Serial.begin(9600);
Wire.begin();                      // Initialize comunication
Wire.beginTransmission(MPU);       // Start communication with MPU6050
//The 0x6B register is the power management register, and
//writing a value of 0x00 to it sets the device to use its internal oscillator as the clock source.
Wire.write(0x6B);
Wire.write(0x00);
Wire.endTransmission(true);        //end the transmission
```

To calibrate the sensor, we take the average of the first 200 values for acceleration and angular velocity (the sensor is kept completely stationary during this time) and subtract these error values from the data collected afterwards. In the code calculate_error() function performs this task.

Code for MPU6050:-

```
// === Read accelerometer data === //
Wire.beginTransmission(MPU);
Wire.write(0x3B); // Start with register 0x3B (ACCEL_XOUT_H)
Wire.endTransmission(false);

Wire.requestFrom(MPU, 6, true);

AccX = ((Wire.read() << 8 | Wire.read()) / 16384.0) - AccErrorX; // X-axis value
AccY = ((Wire.read() << 8 | Wire.read()) / 16384.0) - AccErrorY; // Y-axis value
AccZ = ((Wire.read() << 8 | Wire.read()) / 16384.0) - AccErrorZ; // Z-axis value
```

```
// === Read gyroscope data === //
Wire.beginTransmission(MPU);
Wire.write(0x43);
Wire.endTransmission(false);

Wire.requestFrom(MPU, 6, true);
// For a 250deg/s range we have to divide first the raw value by 131.0, according to the datasheet
//for the ±250 DPS range, one LSB represents 1/131 degrees per second
GyroX = ((Wire.read() << 8 | Wire.read()) / 131.0) - GyroErrorX;
GyroY = ((Wire.read() << 8 | Wire.read()) / 131.0) - GyroErrorY;
GyroZ = ((Wire.read() << 8 | Wire.read()) / 131.0) - GyroErrorZ;
```

Code for GPS Module:-

```
// Libraries used for GPS Module
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
```

```
// Here we make pin 4 as RX of arduino & pin 3 as TX of arduino
static const int RXPin = 4, TXPin = 3;
volatile int degree, secs, mins;
volatile float minutes, seconds;
TinyGPSPlus gps;
SoftwareSerial ss(RXPin, TXPin);
```

```
void setup()
{
  Serial.begin(9600);
  ss.begin(9600);
}
```

Code for GPS Module:-

```
while (ss.available() > 0)
  // display the information when connection is established with the gps module
  if (gps.encode(ss.read()))
    displayInfo();

  // if the coonection isn't established for 5 secs then printing the error message
  if (millis() > 5000 && gps.charsProcessed() < 10)
  {
    Serial.println(F("No GPS detected: check wiring."));
    while(true);
  }
```

```
// fucntion used to get data for date
String(date) = (String(gps.date.day()) + "/" + String(gps.date.month()) + "/" + String(gps.date.year()));
```

```
// fucntion used to get data for time
String(time) = (String(gps.time.hour()) + ":" + String(gps.time.minute()) + ":" + String(gps.time.second()));
```

```
// fucntion used to get data for latitude
lat_val = gps.location.lat();
```

```
// fucntion used to get data for longitude
lng_val = gps.location.lng();
```

Code for SD Card:-

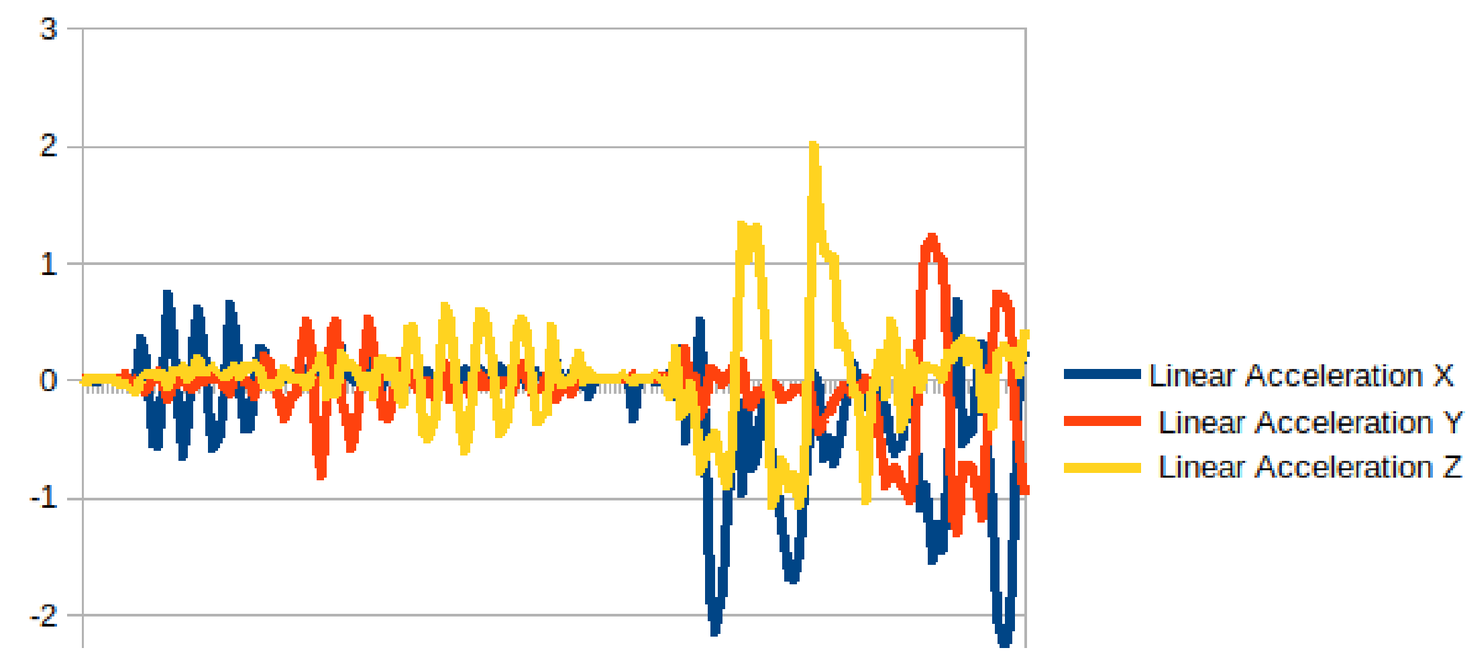
```
// include the SD library:
#include <SPI.h>
#include <SD.h>
```

```
File myFile;
const int chipSelect = 10;
```

```
if (SD.begin()){
    Serial.println("SD card is ready to use.");
}else
{
    Serial.println("SD card initialization failed");
    return;
}
```

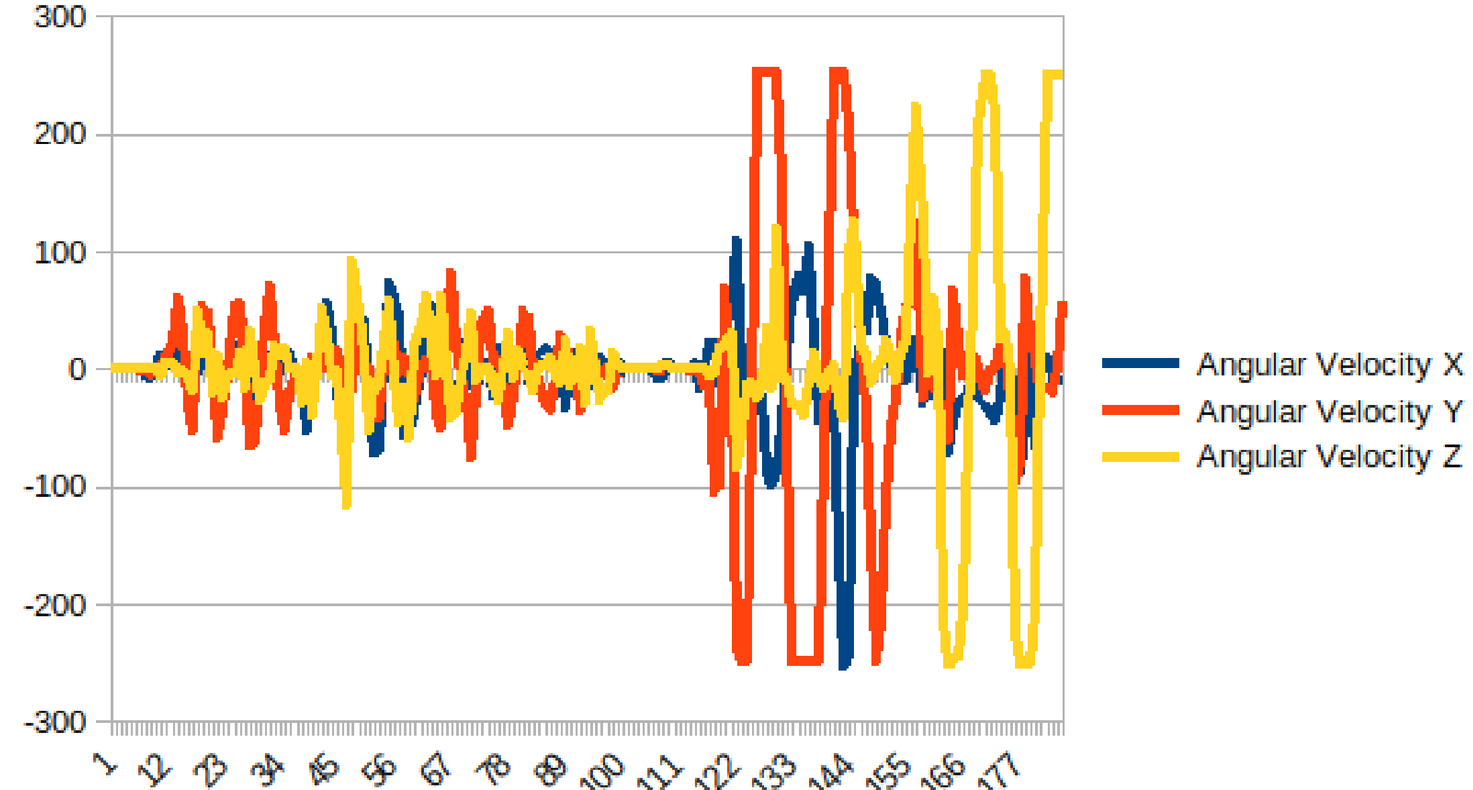
```
// write the accelerometer values to SD card
  myFile = SD.open("data.txt", FILE_WRITE);
  if (myFile) {
    String(datastr) = (date + time + lat_val + latitude + lng_val + longitude + AccX + String(", ") + AccY
                      + String(", ") + AccZ + String(", ") + GyroX + String(", ") + GyroY + String(", ") + GyroZ);
    myFile.println(datastr);

    myFile.close();
  }
```

Test Data



Linear Accelaration

1. Moved the circuit along X-Axis
2. Then Along Y-axis
3. Along Z-axis



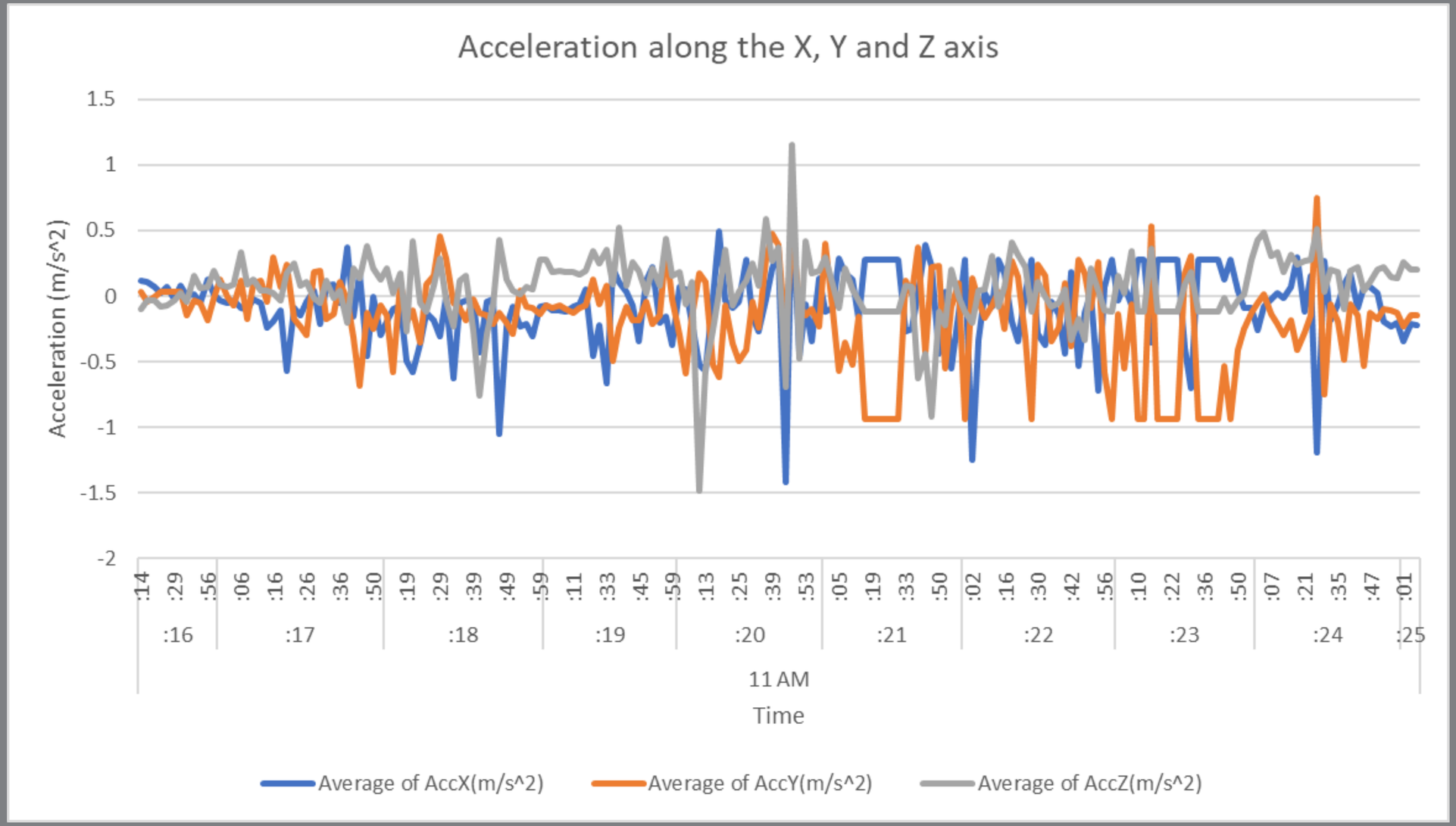
Angular Velocity

1. Rotated about Y-axis
2. Then Along Z-axis

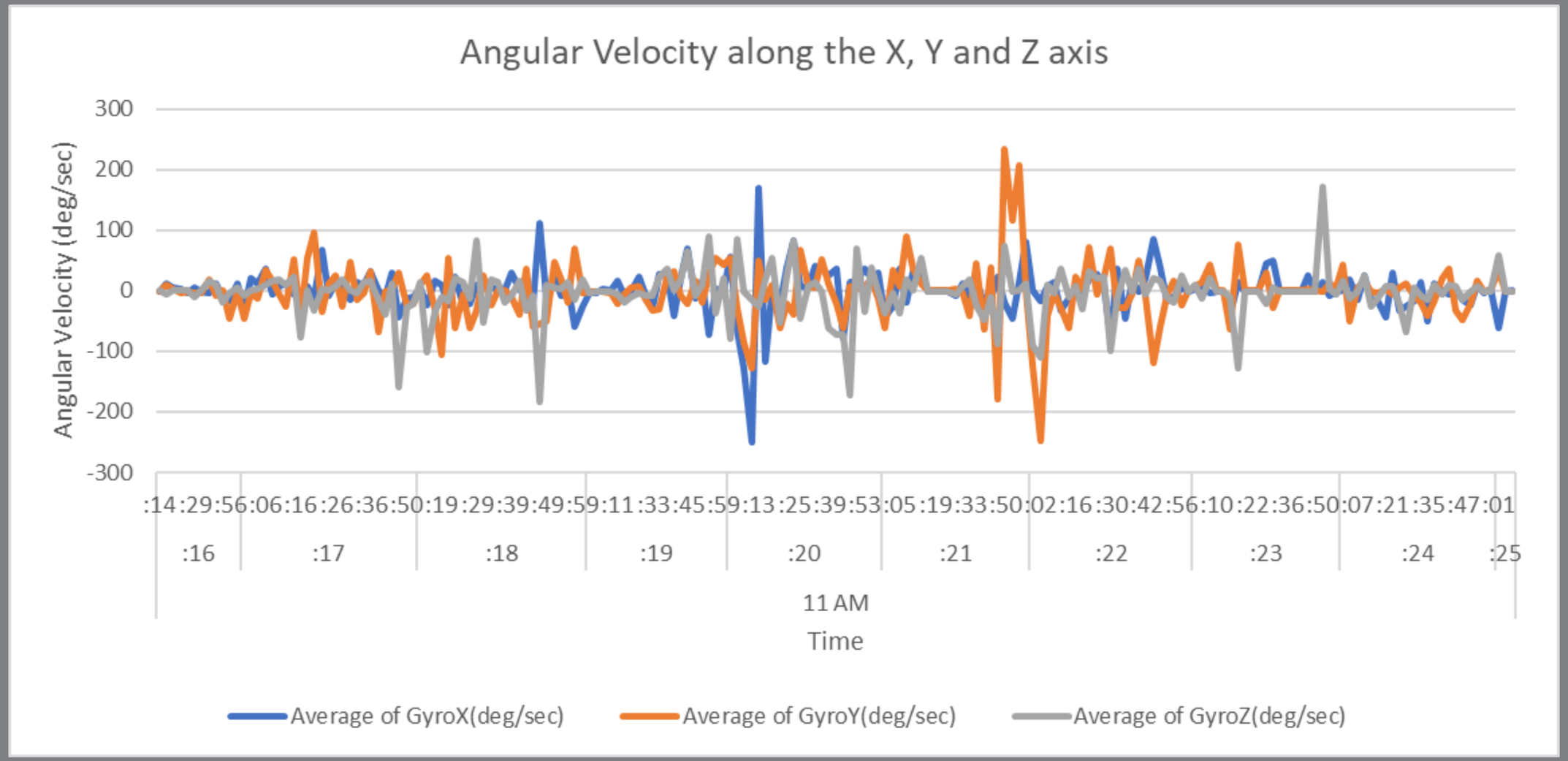
Data Collected is exported to Excel

Date	Time	Latitude(dec)	Latitude(deg min sec)	Longitude(dec)	Longitude(deg min sec)	AccX(m/s^2)	AccY(m/s^2)	AccZ(m/s^2)	GyroX(deg/sec)	GyroY(deg/sec)	GyroZ(deg/sec)
25/3/2023	11:16:14 AM	17.445558	17°26'44N	78.348175	78°20'53E	0.11	0.04	-0.12	-1.16	-1.08	-0.41
25/3/2023	11:16:14 AM	17.445558	17°26'44N	78.348175	78°20'53E	0.12	0.03	-0.12	-0.37	-0.6	-0.15
25/3/2023	11:16:14 AM	17.445558	17°26'44N	78.348175	78°20'53E	0.12	0.04	-0.1	-0.4	-0.91	-0.46
25/3/2023	11:16:14 AM	17.445558	17°26'44N	78.348175	78°20'53E	0.13	0.04	-0.05	0.02	-5.96	0.8
25/3/2023	11:16:21 AM	17.445558	17°26'44N	78.348175	78°20'53E	0.11	-0.04	-0.03	12.31	9.42	-6.23
25/3/2023	11:16:23 AM	17.445558	17°26'44N	78.348175	78°20'53E	0.07	0	-0.03	4.37	2.24	0.99
25/3/2023	11:16:25 AM	17.445558	17°26'44N	78.348175	78°20'53E	0.02	0.03	-0.08	2.05	-4.08	0.96
25/3/2023	11:16:27 AM	17.445558	17°26'44N	78.348175	78°20'53E	0.07	0.03	-0.07	-3.17	-0.32	0.51
25/3/2023	11:16:29 AM	17.445558	17°26'44N	78.348175	78°20'53E	-0.02	0.03	-0.03	5.37	-4.21	-10.59
25/3/2023	11:16:31 AM	17.445558	17°26'44N	78.348175	78°20'53E	0.08	0.03	0.02	-1.79	-3.54	0.56
25/3/2023	11:16:35 AM	17.445627	17°26'44N	78.348182	78°20'53E	0.11	-0.22	-0.06	-25.05	27.03	12.04
25/3/2023	11:16:35 AM	17.445627	17°26'44N	78.348182	78°20'53E	0.14	-0.23	-0.14	3.47	0.46	26.38
25/3/2023	11:16:35 AM	17.445627	17°26'44N	78.348182	78°20'53E	-0.24	0	0.08	11.96	30.05	8.73
25/3/2023	11:16:42 AM	17.445627	17°26'44N	78.348182	78°20'53E	0	0.15	0.24	-0.48	-25.95	-7.64
25/3/2023	11:16:42 AM	17.445627	17°26'44N	78.348182	78°20'53E	0.07	-0.27	0.05	14.62	16.19	31.1
25/3/2023	11:16:42 AM	17.445627	17°26'44N	78.348182	78°20'53E	-0.04	0.03	0.19	19.57	-15.79	5.61
25/3/2023	11:16:49 AM	17.445627	17°26'44N	78.348182	78°20'53E	-0.18	0.12	0.06	-24.11	43.85	-13.59
25/3/2023	11:16:49 AM	17.445627	17°26'44N	78.348182	78°20'53E	-0.01	0.11	0.08	-11.31	2.82	-23.26

INITIAL INSTANTANEOUS DATA PLOT OBTAINED:



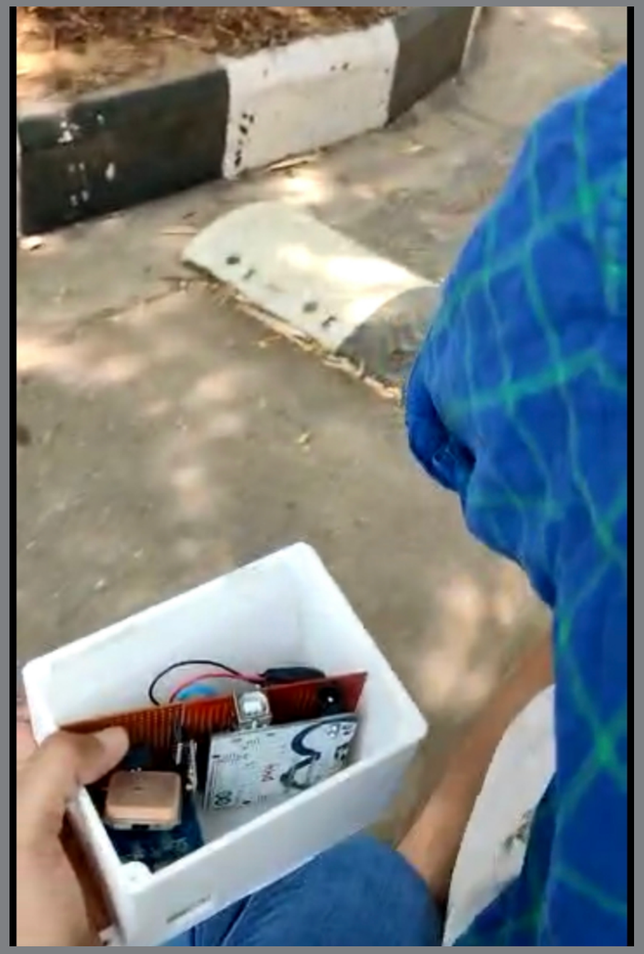
INITIAL INSTANTANEOUS DATA PLOT OBTAINED:



FINAL DATA COLLECTION AND ANALYSIS

DATA COLLECTION

26 - 03 - 23



DATA COLLECTION VIDEO LINK:

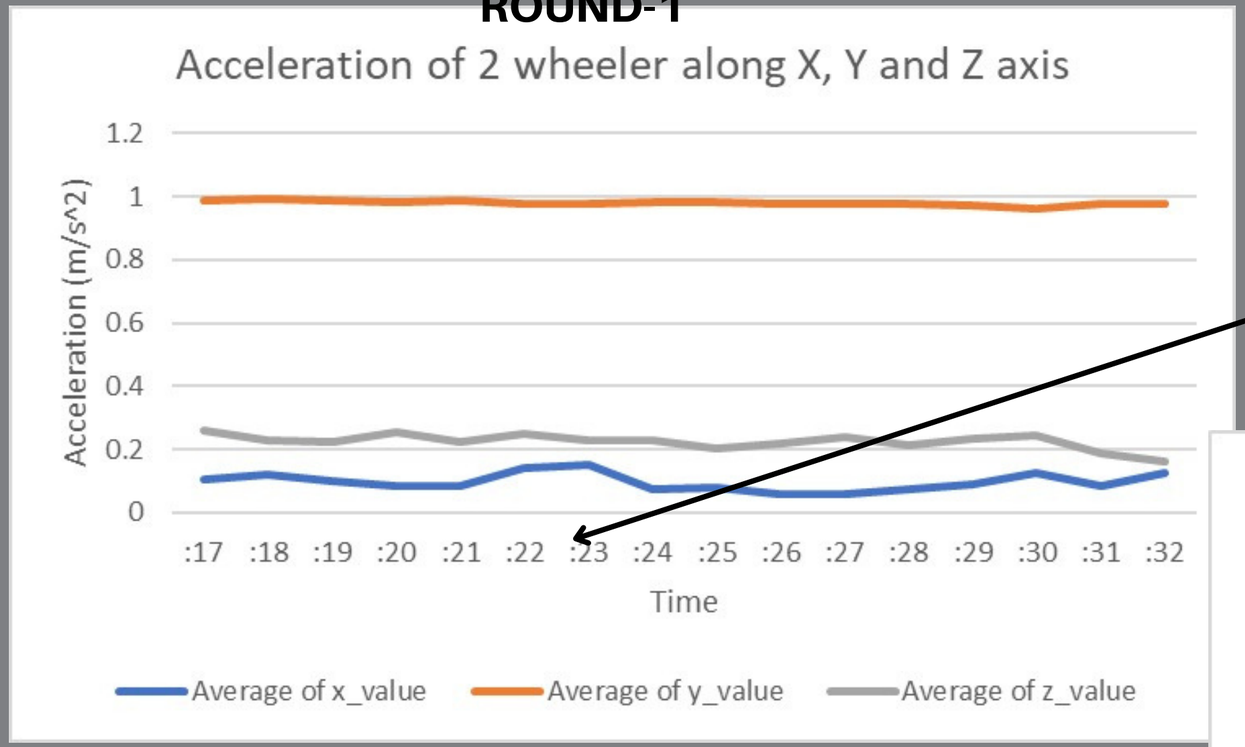
https://iiitaphyd-my.sharepoint.com/:v:/g/personal/pabba_ananya_students_iiit_ac_in/EatdgEhpn6pKqo-NTVitx8sBWZG2XvSMn_siUX4oXWytFw?e=E8dHqH

INITIAL DATA COLLECTION SCREEN RECORDING LINK:

https://iiitaphyd-my.sharepoint.com/:v:/g/personal/pabba_ananya_students_iiit_ac_in/Ed6_Xvwf27hHtG9_1KOA0_IBHWn5LnqEHdkCsFPbOh_nzw?e=dadyvT

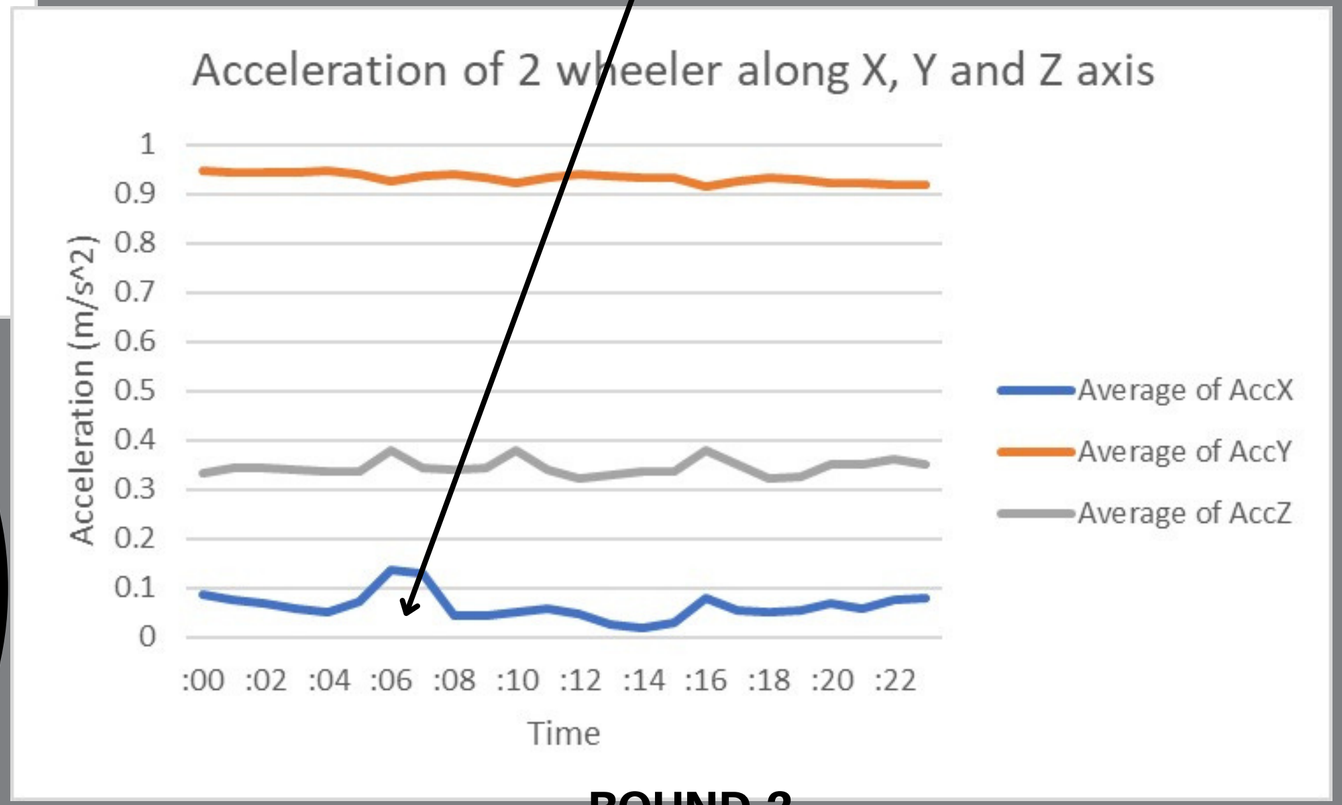
FINAL DATA ANALYSIS OF TWO SAME-ROUTE ROUNDS OF CAMPUS:

ROUND-1



Similar pattern observed
for both rounds at the same location as we moved with a higher speed, so acceleration increased, then maintained that acceleration and then decreased speed due to which acceleration also decreased

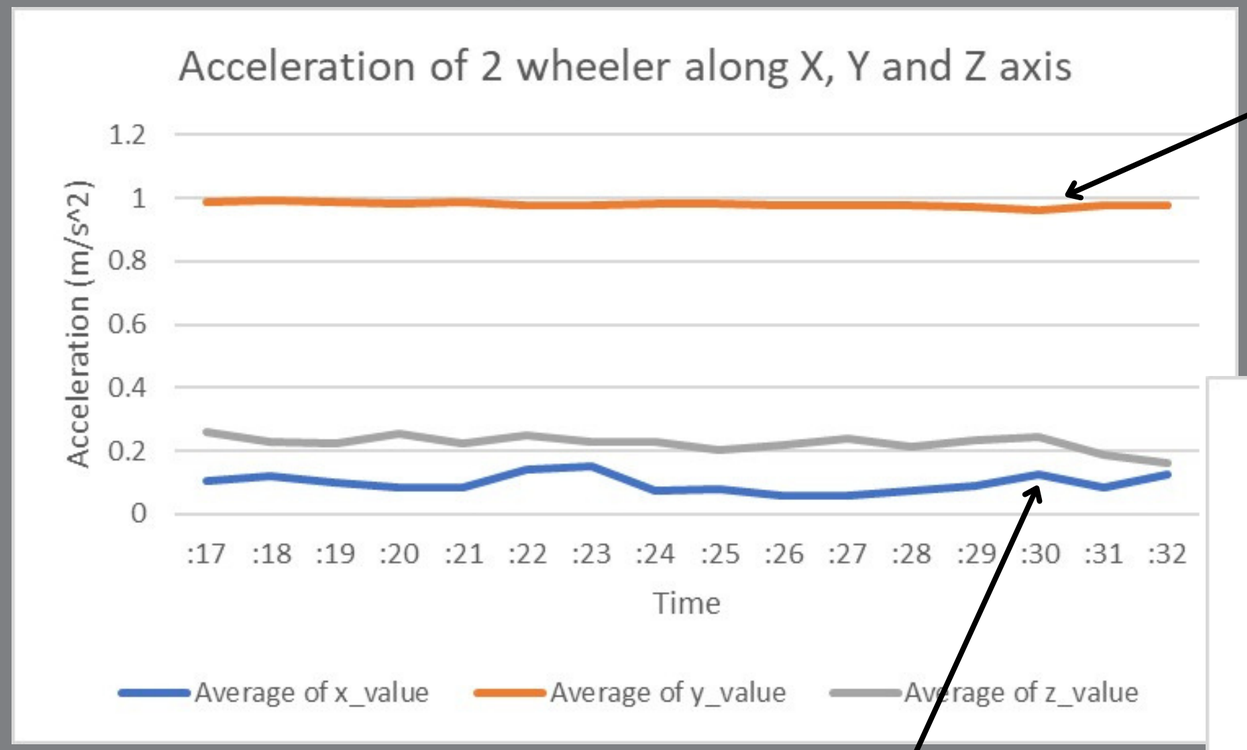
X- axis represents left-right direction
Y-axis represents vertical (up-down) direction
Z-axis reprensents the forward direction
One unit in the y-axis of the plots is equivalent to 1g m/s^2 (approx 10 m/s^2)
We get a almost constant acceleration of 1g m/s^2 in vertical direction due to acceleration due to gravity.
This shows that our sensor is properly callibrated.



ROUND-2

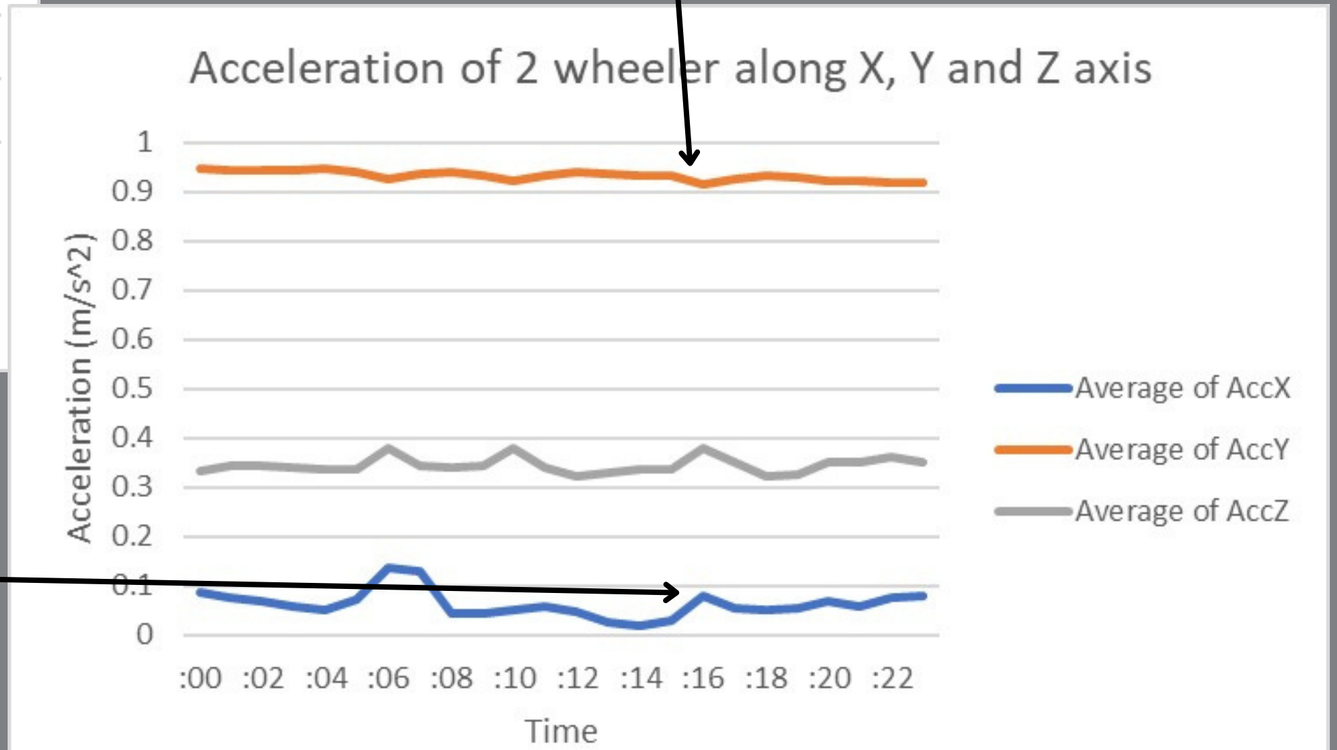
FINAL DATA ANALYSIS OF TWO SAME-ROUTE ROUNDS OF CAMPUS:

ROUND-1



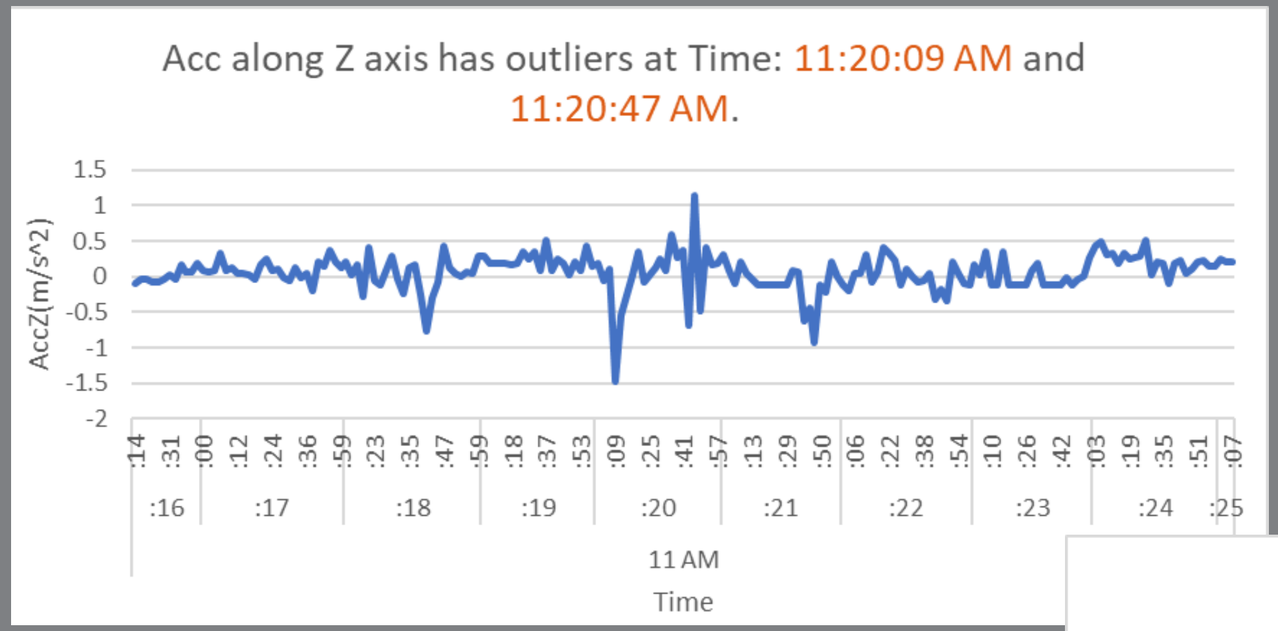
We see a slight decline/drop in the acceleration here where we have a speed breaker/bump in both rounds. Since we have an upside bump, the bike tends to move upwards whereas acceleration due to gravity is downward due to which there is a slight decrease in our acceleration in the up-down direction due to opposite accelerations.

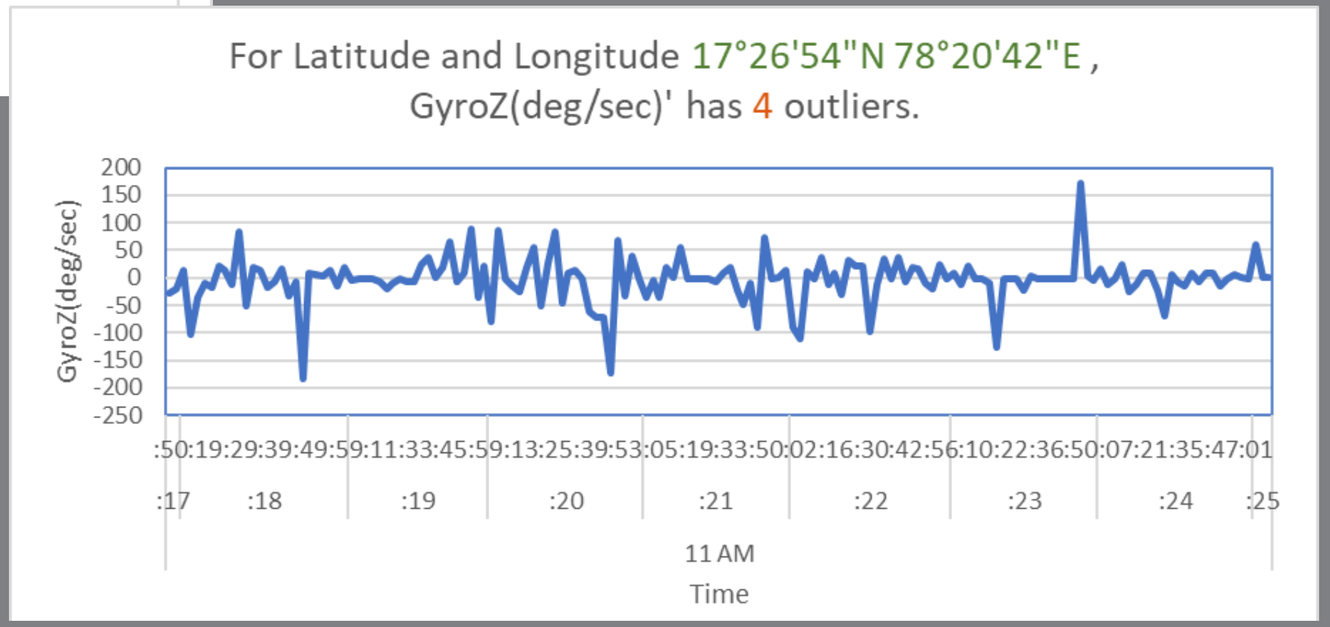
Similar peak observed on both rounds at almost the same time because at that position, in the X-direction, we have increased the acceleration causing incline of the graph and then suddenly decreased it due to a speed breaker/upside bump



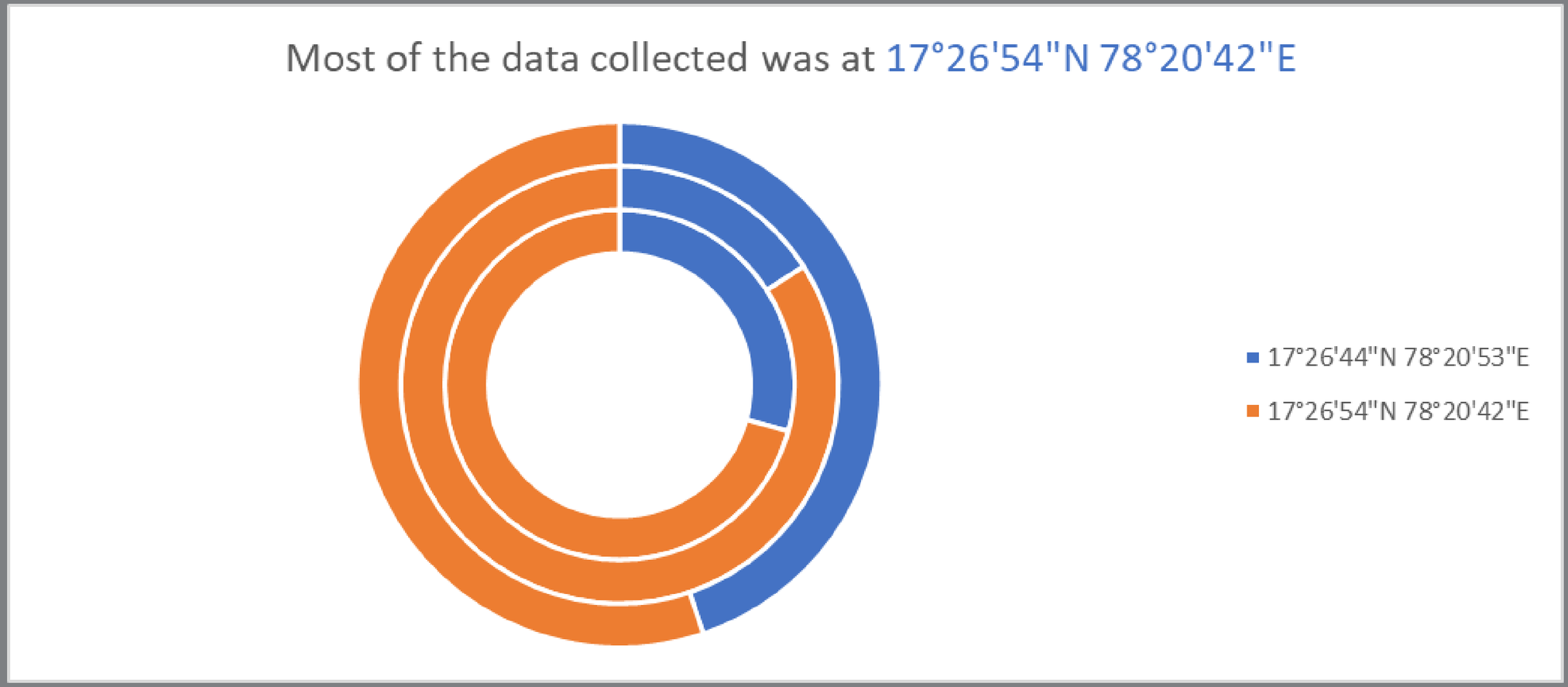
ROUND-2

DATA ANALYSIS:





DATA ANALYSIS:



CHALLENGES FACED:-

- Getting a reliable connection from the satellite to the GPS Module.
- Writing GPS Module results to an SD Card.
- Taking huge amounts of GPS module results and deriving conclusions/data analysis from that.
- Tried working with ESP32, but neo-6m does't connect with ESP32
- Tried to log data to Dashboard using Blink, which supports ESP32, but because of point 2, we couldn't make it happen.

IMPROVEMENTS:-

- Creating a mobile app/website to see the real-time values and changes in the data.
- Using data logged to train ML/RL models that could review road quality.
- Implementing the applications we have mentioned at the beginning of the presentation.
- Building a platform to analyse the data collected and draw desirable conclusions.
- Building software to be integrated into the two-wheelers that can detect overspeeding based on real-time data and notify the rider.

Contribution

Arya Marda
Worked on the hardware implementation of the circuit, collected data and analysis of GPS data and its plotting.

Meghana Tedla
Worked on Arduino code, collecting data, and analysis of the data.

Anushka Agrawal
Worked on Arduino code and collected data.

Ananya Vaibhavi
Worked on the hardware implementation of the circuit, 3D-Design of the Box and collecting data.

THANK YOU !

Our Team:

Arya Marda: 2021102021

Meghana Tedla: 2021102002

Anushka Agrawal: 2021102023

Ananya Vaibhavi: 2021102003