CS – 322
MINI PROJECT REPORT

MOTION CLASSIFICATION USING DEEP LEARNING ON ESP32

Team members:

Rohit Ranjan(2001CS56)

Kartik Mouli (2001CS35)

Report by:

Rohit Ranjan(2001CS56)

Aim of this project:

The aim of this project is to categorize the motion into the different class using Neural Network on ESP32.

About:

Motion Classification is the one of the applications of what the machine learning can do. We have categorized the motion into the 4 classes. They are:

i. Idle
ii. Up Down
iii. Left Right
iv. Circle

Equipment Used:

1. **ESP32 WIFI Module**

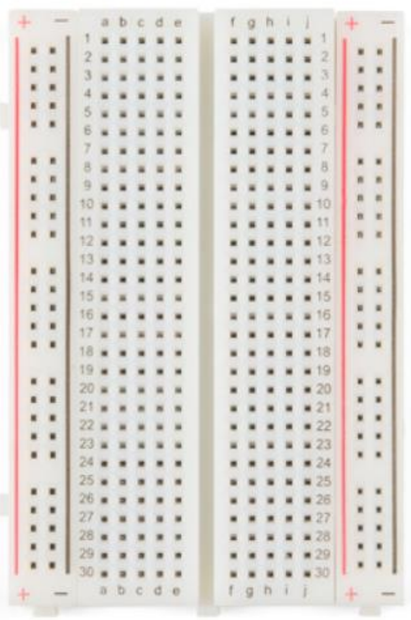
   ESP32 is a series of low-cost, low-power system on a chip microcontroller with integrated WIFI and Bluetooth radio module.

2. **MPU 6050 – 6 DOF Sensor**

   It consists of 3-Axis Accelerometer and Gyro Sensor on the same silicon together with an onboard Digital Motion Processor capable of processing complex 9 axis Motion Fusion Algorithms.

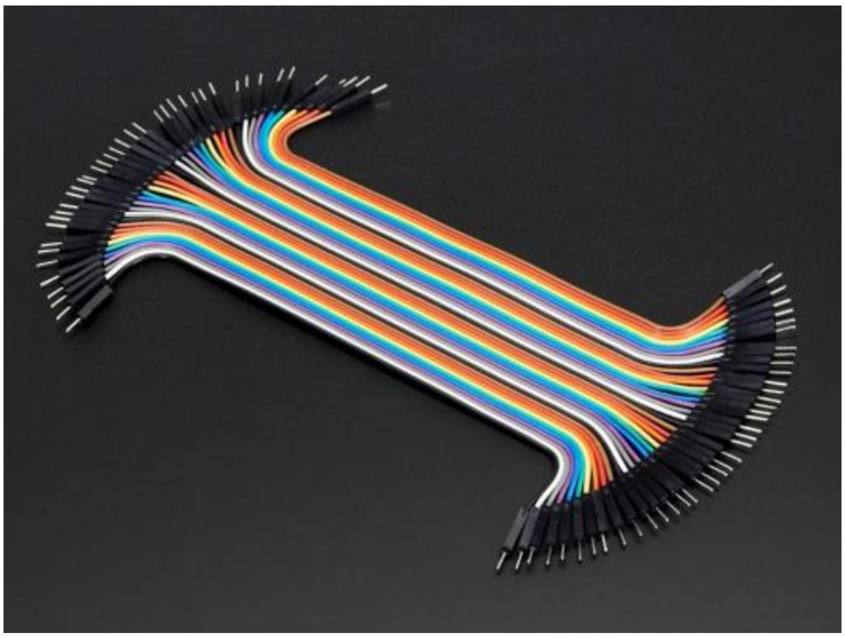
3. **Breadboard**

   A breadboard is a construction base used to build semi-permanent prototypes of electronic circuits.



4. **Jumper Wires**

   A jumper wire is an electrical wire, or group of them in a cable, with a connector or pin at each end which is normally used to interconnect the components of a breadboard.

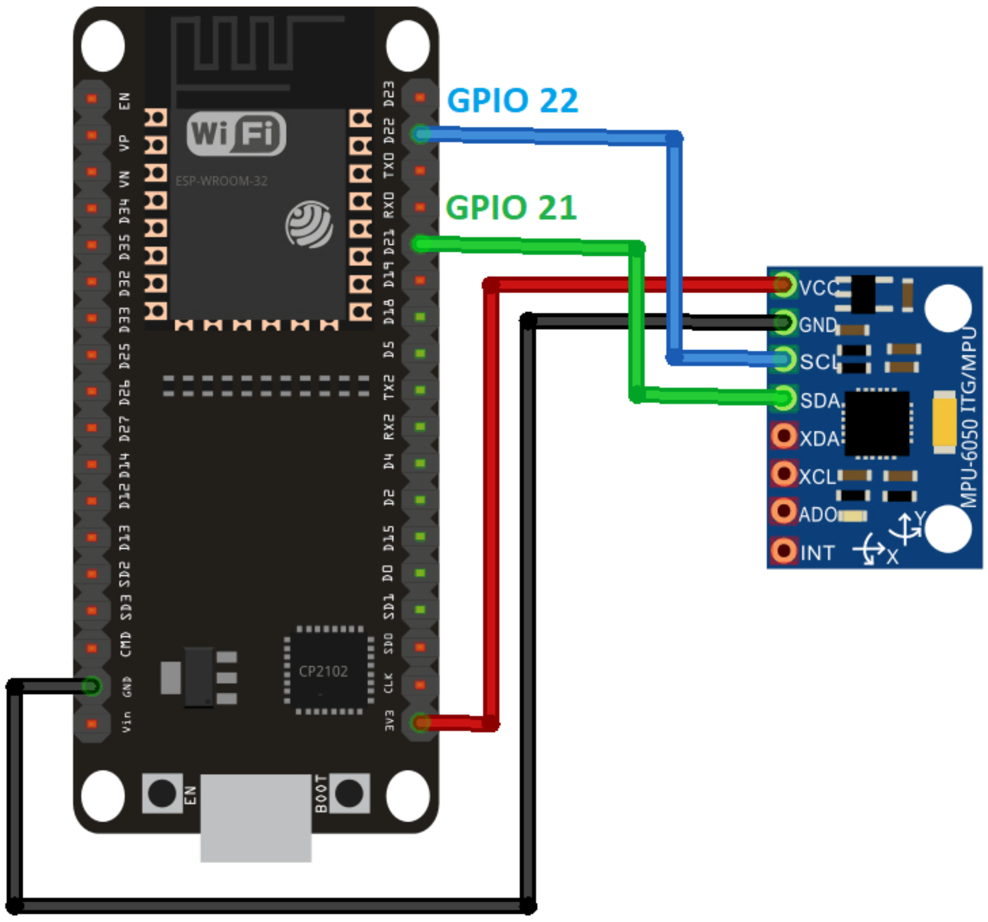


Software and Online Services used:

1. **Edge Impulse Studio:**
   Edge Impulse provides the ultimate development experience for ML on embedded devices for sensors, audio, and computer vision, at scale. It enables the deployment of highly-optimized ML on hardware ranging from MCUs to CPUs and custom AI accelerators.

2. **Arduino IDE:**
   It is an IDE used to write the code and upload it on the board.

Schematic



Hardware Connections:

ESP32	–>	MPU 6050
D22 (GIPO 22)	–>	SCL
D21 (GPIO 21)	–>	SDA
3V3	–>	VCC
GND	–>	GND

Project Contribution:

The project has been divided into the 4 parts. They are:

1. Testing hardware components
2. Data Acquisition
3. Creating ML model
4. Deployment of model

I have done part 1 and 2 and the rest has been handled by Kartik Mouli.

Working:

The MPU 6050 sensor collects the data in each and every second interval and send over to ESP32 module which has motion classification model deployed on it. It classifies the motion and gives the output on serial monitor.

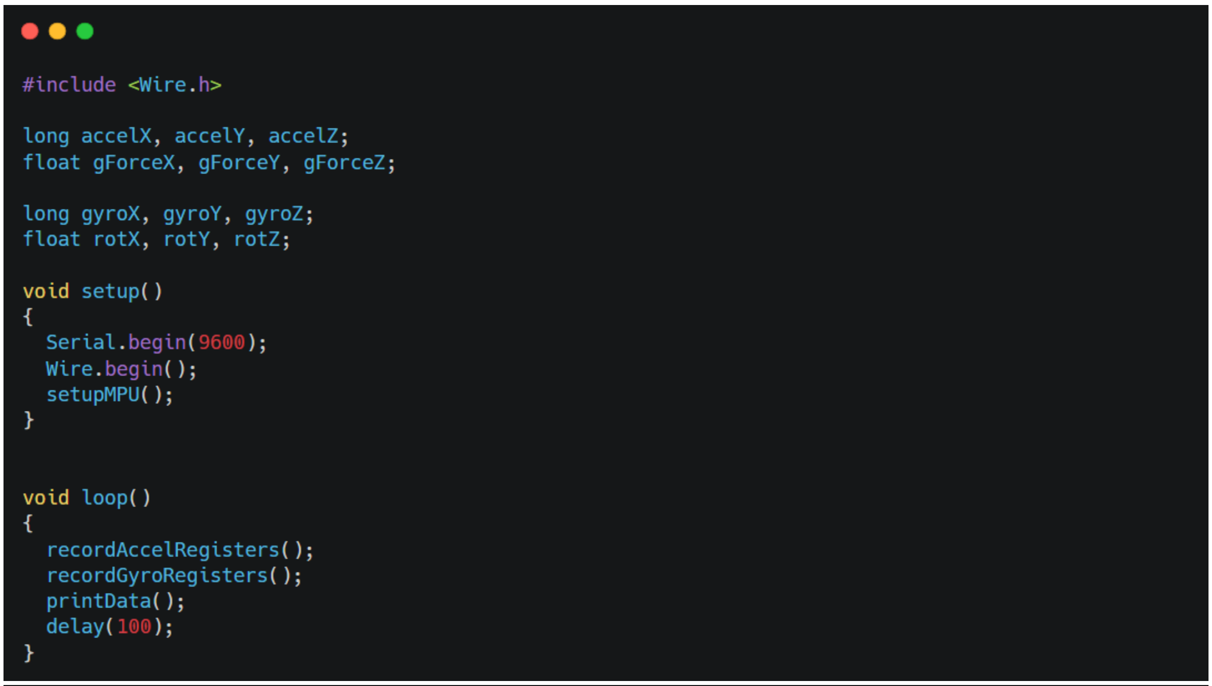
We have used Edge Impulse Studio which provides great environment for data acquisition and creating ML model. We deployed the model on ESP32 using Arduino IDE.

1. **Testing hardware components**

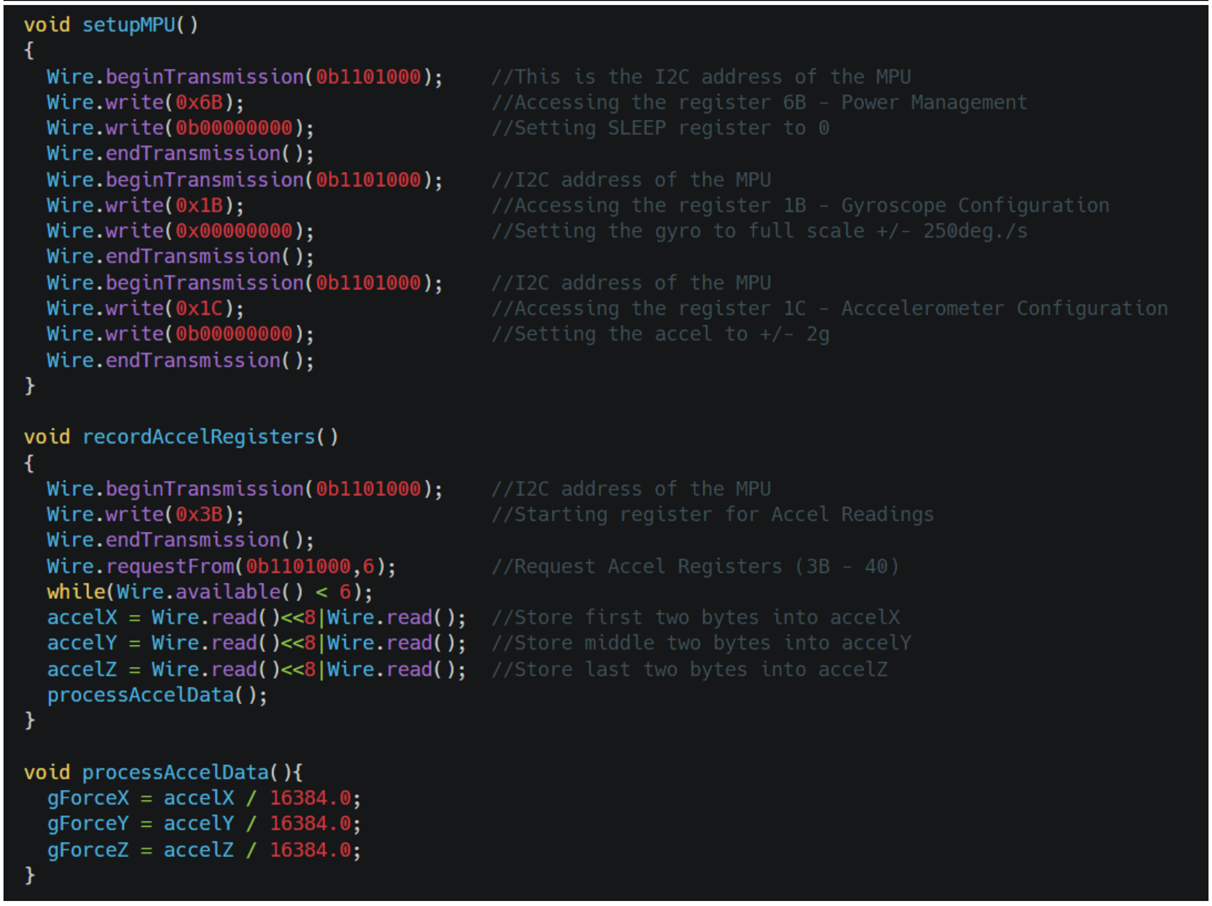
MPU 6050:	6 axis sensor or 6 degree of freedom sensor Both accelerometer and gyro meter sensors are present in this single module. Accelerometer sensor give output readings in this term of force applied on object due to gravity and gyro meter sensor give output in terms of angular displacement of the object in clockwise or anticlockwise direction.

The following is the code to test accelerometer and gyro meter.

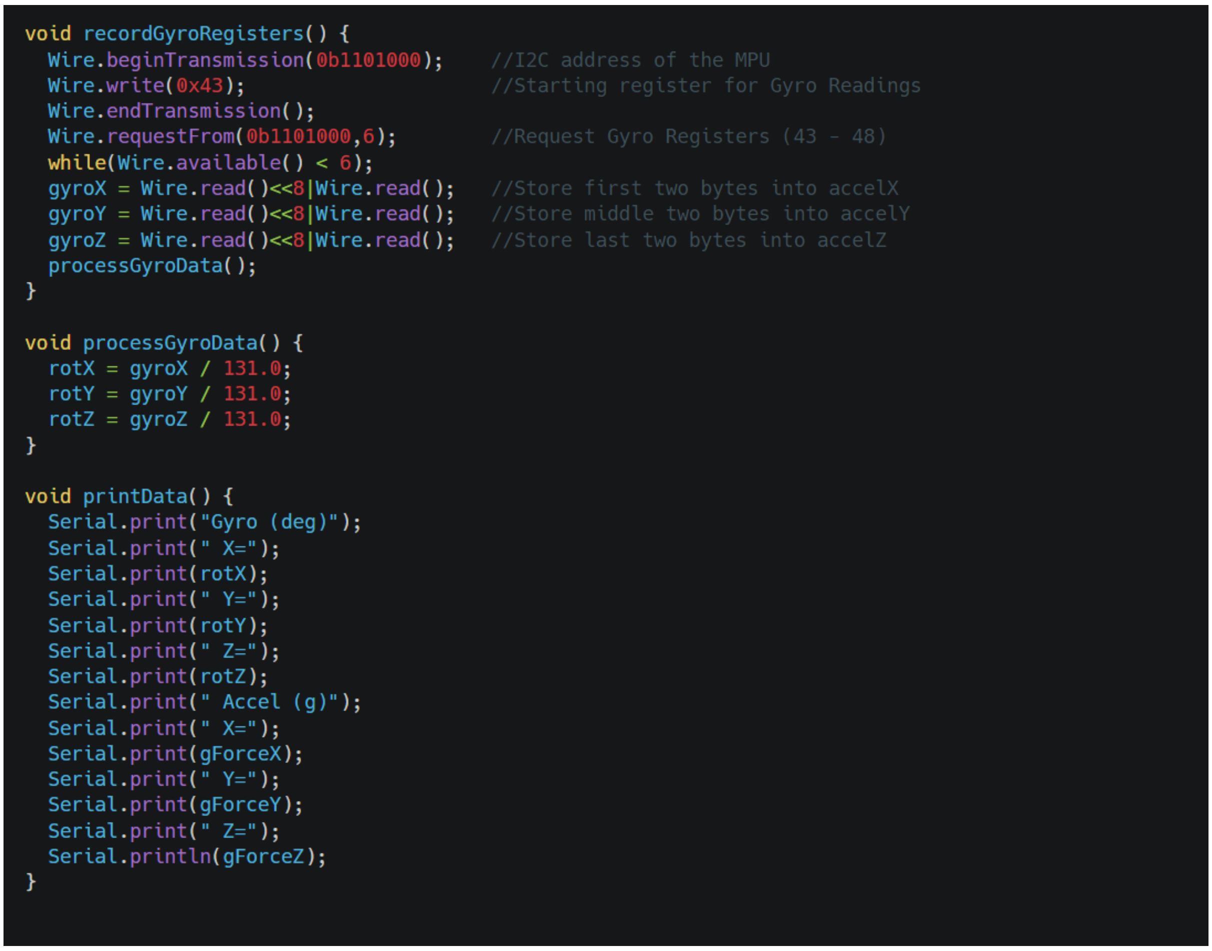
MPU6050 sensor use SCL and SDA line of ESP32 DEVKIT V1, therefore, we will be using Wire.h library in the code for I2C communication.



```
#include <Wire.h>

long accelX, accelY, accelZ;
float gForceX, gForceY, gForceZ;

long gyroX, gyroY, gyroZ;
float rotX, rotY, rotZ;

void setup()
{
  Serial.begin(9600);
  Wire.begin();
  setupMPU();
}


void loop()
{
  recordAccelRegisters();
  recordGyroRegisters();
  printData();
  delay(100);
}
```



```
void setupMPU()
{
  Wire.beginTransmission(0b1101000);     //This is the I2C address of the MPU
  Wire.write(0x6B);                      //Accessing the register 6B - Power Management
  Wire.write(0b00000000);                //Setting SLEEP register to 0
  Wire.endTransmission();
  Wire.beginTransmission(0b1101000);     //I2C address of the MPU
  Wire.write(0x1B);                      //Accessing the register 1B - Gyroscope Configuration
  Wire.write(0x00000000);                //Setting the gyro to full scale +/- 250deg./s
  Wire.endTransmission();
  Wire.beginTransmission(0b1101000);     //I2C address of the MPU
  Wire.write(0x1C);                      //Accessing the register 1C - Acccelerometer Configuration
  Wire.write(0b00000000);                //Setting the accel to +/- 2g
  Wire.endTransmission();
}

void recordAccelRegisters()
{
  Wire.beginTransmission(0b1101000);     //I2C address of the MPU
  Wire.write(0x3B);                      //Starting register for Accel Readings
  Wire.endTransmission();
  Wire.requestFrom(0b1101000,6);         //Request Accel Registers (3B - 40)
  while(Wire.available() < 6);
  accelX = Wire.read()<<8|Wire.read();   //Store first two bytes into accelX
  accelY = Wire.read()<<8|Wire.read();   //Store middle two bytes into accelY
  accelZ = Wire.read()<<8|Wire.read();   //Store last two bytes into accelZ
  processAccelData();
}

void processAccelData(){
  gForceX = accelX / 16384.0;
  gForceY = accelY / 16384.0;
  gForceZ = accelZ / 16384.0;
}
```



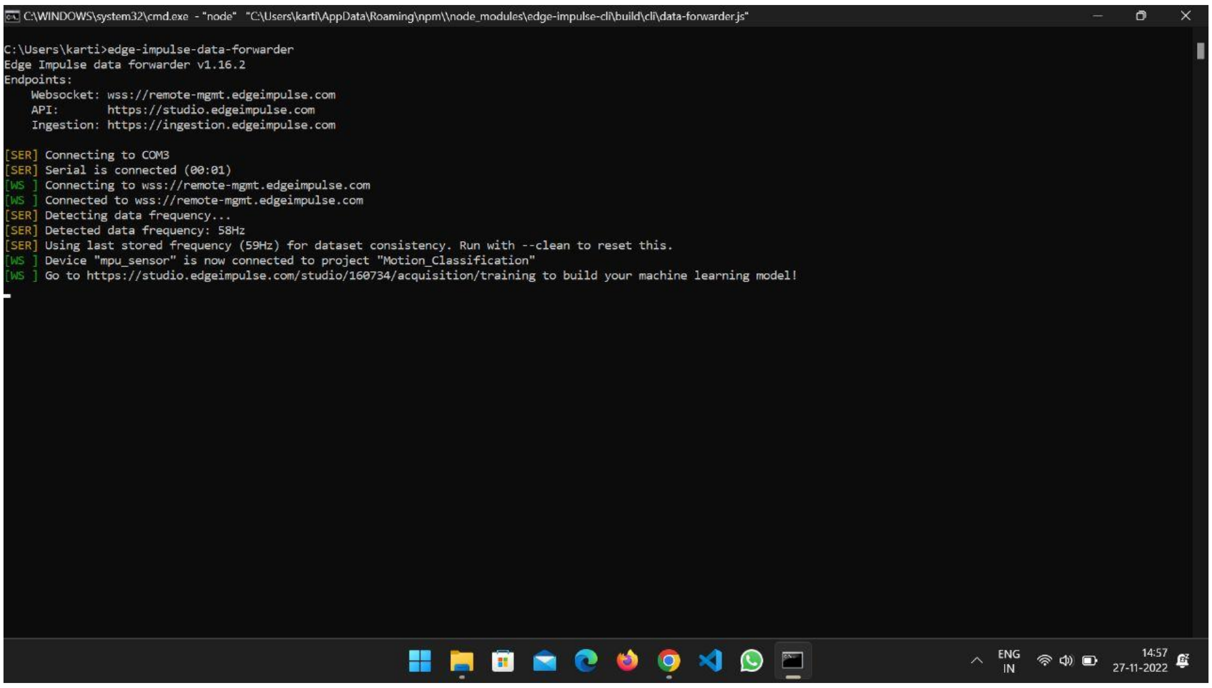
```
void recordGyroRegisters() {
  Wire.beginTransmission(0b1101000);    //I2C address of the MPU
  Wire.write(0x43);                     //Starting register for Gyro Readings
  Wire.endTransmission();
  Wire.requestFrom(0b1101000,6);        //Request Gyro Registers (43 - 48)
  while(Wire.available() < 6);
  gyroX = Wire.read()<<8|Wire.read();   //Store first two bytes into accelX
  gyroY = Wire.read()<<8|Wire.read();   //Store middle two bytes into accelY
  gyroZ = Wire.read()<<8|Wire.read();   //Store last two bytes into accelZ
  processGyroData();
}

void processGyroData() {
  rotX = gyroX / 131.0;
  rotY = gyroY / 131.0;
  rotZ = gyroZ / 131.0;
}

void printData() {
  Serial.print("Gyro (deg)");
  Serial.print(" X=");
  Serial.print(rotX);
  Serial.print(" Y=");
  Serial.print(rotY);
  Serial.print(" Z=");
  Serial.print(rotZ);
  Serial.print(" Accel (g)");
  Serial.print(" X=");
  Serial.print(gForceX);
  Serial.print(" Y=");
  Serial.print(gForceY);
  Serial.print(" Z=");
  Serial.println(gForceZ);
}
```

2. Data Acquisition

Data Forwarder: It is a tool through which one can uploads the data available in the serial monitor to the Edge Impulse.

Arduino Sketch is used to uploads data to the serial with some sample frequency and with “,” to separate the data.

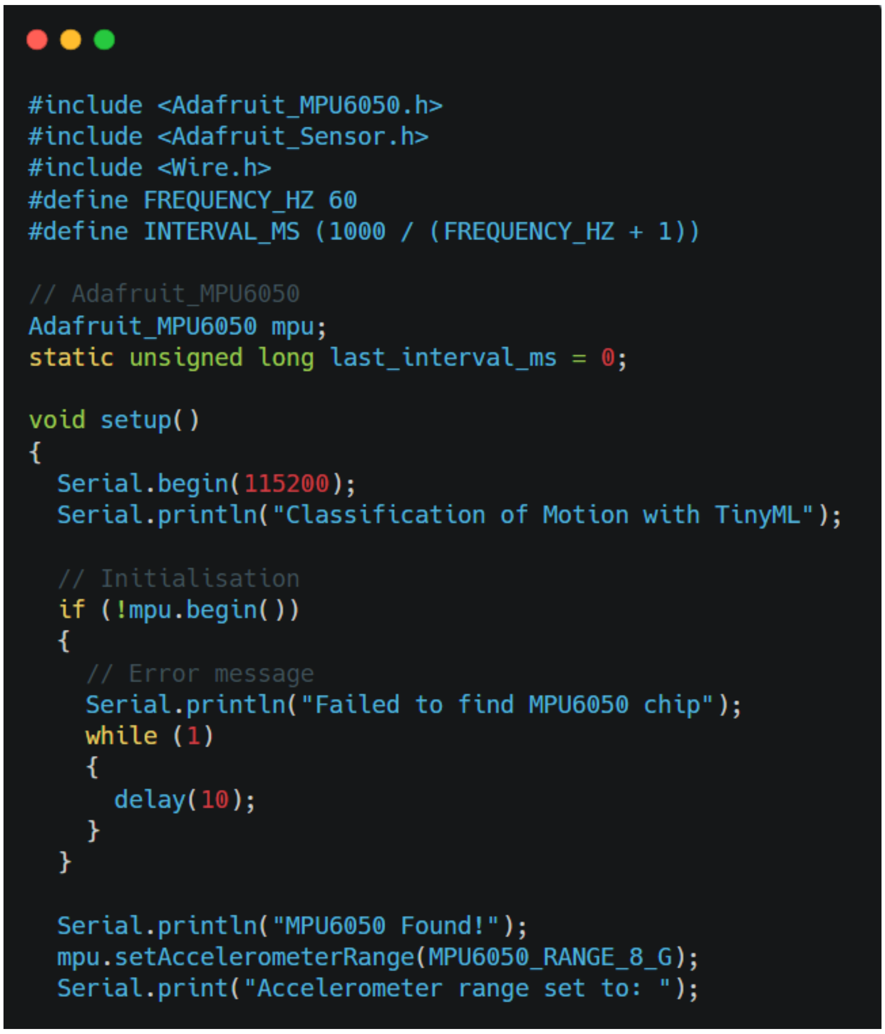
Connecting Edge Impulse account with the CLI.



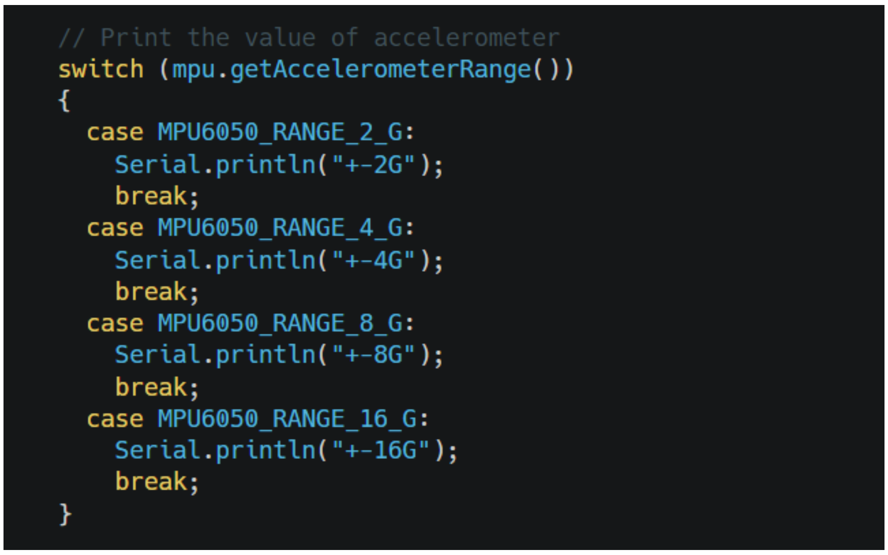
```
C:\WINDOWS\system32\cmd.exe - "node"  "C:\Users\karti\AppData\Roaming\npm\\node_modules\edge-impulse-cli\build\cli\data-forwarder.js"

C:\Users\karti>edge-impulse-data-forwarder
Edge Impulse data forwarder v1.16.2
Endpoints:
    Websocket: wss://remote-mgmt.edgeimpulse.com
    API:       https://studio.edgeimpulse.com
    Ingestion: https://ingestion.edgeimpulse.com

[SER] Connecting to COM3
[SER] Serial is connected (00:01)
[WS ] Connecting to wss://remote-mgmt.edgeimpulse.com
[WS ] Connected to wss://remote-mgmt.edgeimpulse.com
[SER] Detecting data frequency...
[SER] Detected data frequency: 58Hz
[SER] Using last stored frequency (59Hz) for dataset consistency. Run with --clean to reset this.
[WS ] Device "mpu_sensor" is now connected to project "Motion_Classification"
[WS ] Go to https://studio.edgeimpulse.com/studio/160734/acquisition/training to build your machine learning model!
```

Following is the code for data collection.

It will print the value of accelerometer and gyro meter that is read by the MPU 6050.

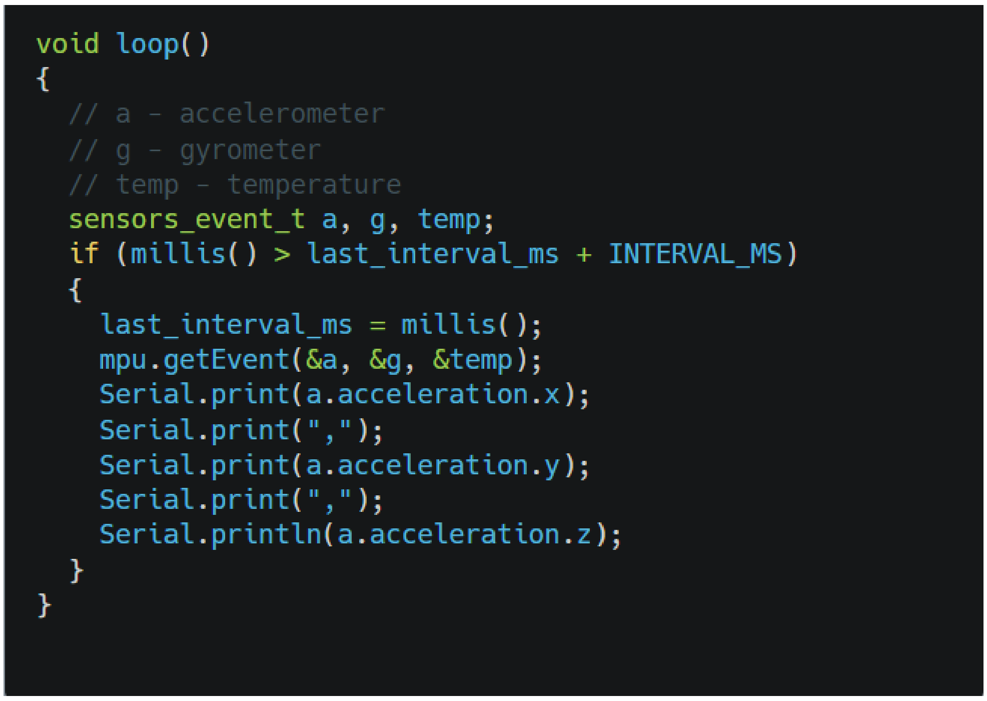


```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>
#define FREQUENCY_HZ 60
#define INTERVAL_MS (1000 / (FREQUENCY_HZ + 1))

// Adafruit_MPU6050
Adafruit_MPU6050 mpu;
static unsigned long last_interval_ms = 0;

void setup()
{
  Serial.begin(115200);
  Serial.println("Classification of Motion with TinyML");

  // Initialisation
  if (!mpu.begin())
  {
    // Error message
    Serial.println("Failed to find MPU6050 chip");
    while (1)
    {
      delay(10);
    }
  }

  Serial.println("MPU6050 Found!");
  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  Serial.print("Accelerometer range set to: ");
```



```
  // Print the value of accelerometer
  switch (mpu.getAccelerometerRange())
  {
    case MPU6050_RANGE_2_G:
      Serial.println("+-2G");
      break;
    case MPU6050_RANGE_4_G:
      Serial.println("+-4G");
      break;
    case MPU6050_RANGE_8_G:
      Serial.println("+-8G");
      break;
    case MPU6050_RANGE_16_G:
      Serial.println("+-16G");
      break;
  }
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  Serial.print("Gyro range set to: ");

  // Print the value of the Gyrometer
  switch (mpu.getGyroRange()) {
    case MPU6050_RANGE_250_DEG:
      Serial.println("+- 250 deg/s");
      break;
    case MPU6050_RANGE_500_DEG:
      Serial.println("+- 500 deg/s");
      break;
    case MPU6050_RANGE_1000_DEG:
      Serial.println("+- 1000 deg/s");
      break;
    case MPU6050_RANGE_2000_DEG:
      Serial.println("+- 2000 deg/s");
      break;
  }

  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
  Serial.print("Filter bandwidth set to: ");
```

```
  // Range of frequencies
  switch (mpu.getFilterBandwidth())
  {
    case MPU6050_BAND_260_HZ:
      Serial.println("260 Hz");
      break;
    case MPU6050_BAND_184_HZ:
      Serial.println("184 Hz");
      break;
    case MPU6050_BAND_94_HZ:
      Serial.println("94 Hz");
      break;
    case MPU6050_BAND_44_HZ:
      Serial.println("44 Hz");
      break;
    case MPU6050_BAND_21_HZ:
      Serial.println("21 Hz");
      break;
    case MPU6050_BAND_10_HZ:
      Serial.println("10 Hz");
      break;
    case MPU6050_BAND_5_HZ:
      Serial.println("5 Hz");
      break;
  }
  Serial.println("");
  delay(100);
}
```



```
void loop()
{
  // a - accelerometer
  // g - gyrometer
  // temp - temperature
  sensors_event_t a, g, temp;
  if (millis() > last_interval_ms + INTERVAL_MS)
  {
    last_interval_ms = millis();
    mpu.getEvent(&a, &g, &temp);
    Serial.print(a.acceleration.x);
    Serial.print(",");
    Serial.print(a.acceleration.y);
    Serial.print(",");
    Serial.println(a.acceleration.z);
  }
}
```

The rest part that is 3 and 4 is covered by Kartik.

---THANK YOU----