Motion & Position Tracking System using MPU6050 sensor

TEAM MEMBERS

- **ALDRICH MARIAN A**
- **BARATH M**
- **ASISH C**
- **AKASH SAI**

Aim:

To design and develop a motion and position tracking system using the MPU6050 sensor to detect orientation, acceleration, and angular velocity of an object in real time.

Components Required:

1. ESP-32
2. MPU-6050
3. Buzzer
4. Breadboard
5. Jumper wire

ESP-32:

ESP32 is a low-cost, low-power system-on-chip (SoC) microcontroller developed by Espressif Systems. It comes with built-in Wi-Fi and Bluetooth connectivity, a dual-core processor, GPIO pins, ADC/DAC, PWM, I²C, SPI, UART interfaces, and is widely used in IoT, automation, robotics, and embedded systems.

MPU-6050:

MPU-6050 is a 6-axis motion tracking device developed by InvenSense. It combines a 3-axis accelerometer and a 3-axis gyroscope on a single chip, with an onboard Digital Motion Processor (DMP) that can process complex 6-axis motion fusion algorithms. It communicates with microcontrollers using the I²C protocol and is widely used in robotics, gaming, gesture recognition, and motion tracking systems.

Buzzer:

A buzzer is an audio signaling device that produces sound when an electric signal is applied, commonly used for alerts, alarms, and notifications in electronic circuits.

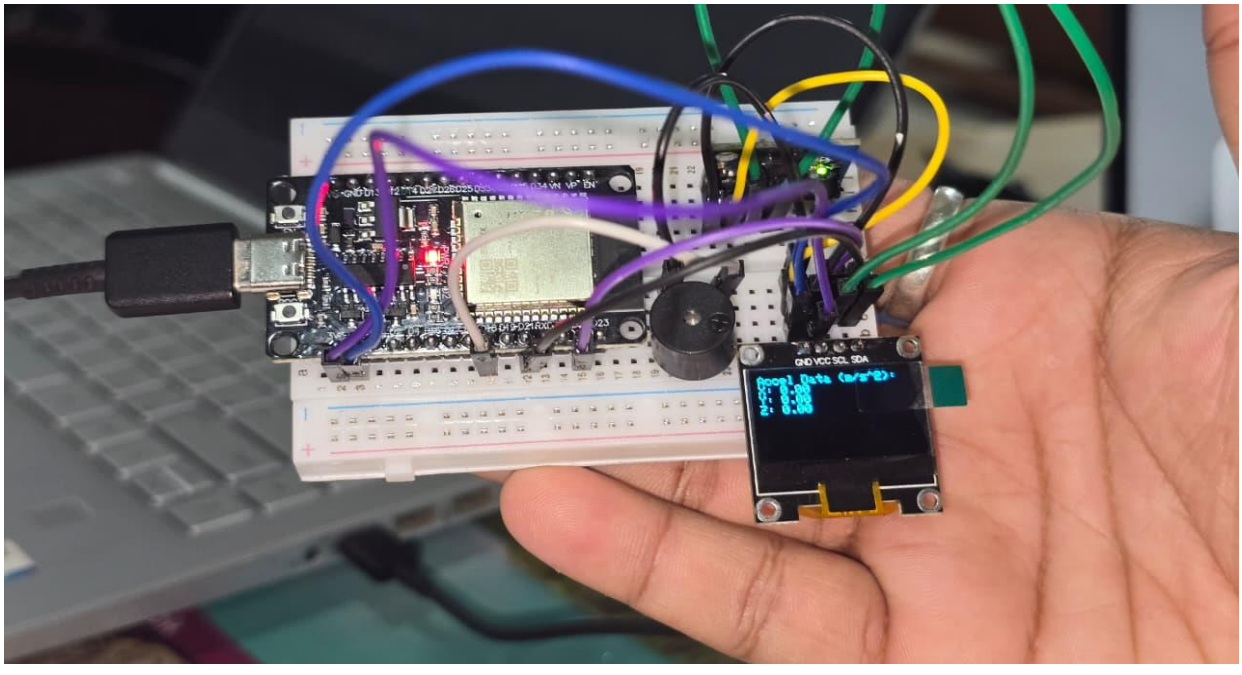
Breadboard:

A breadboard is a reusable electronic prototyping board that allows easy insertion and connection of electronic components without soldering, mainly used for testing and building temporary circuits.

Pin Tables:

Component	Pin on ESP-32	Pin on Sensor
MPU 6050 VCC	3V3	VCC
MPU 6050 Gnd	Gnd	Gnd
MPU 6050 SCL	GPO 22	SCL
MPU 6050 SDA	GPO 21	SDA
Buzzer VCC	GPO 2	VCC
Buzzer Gnd	GND	GND

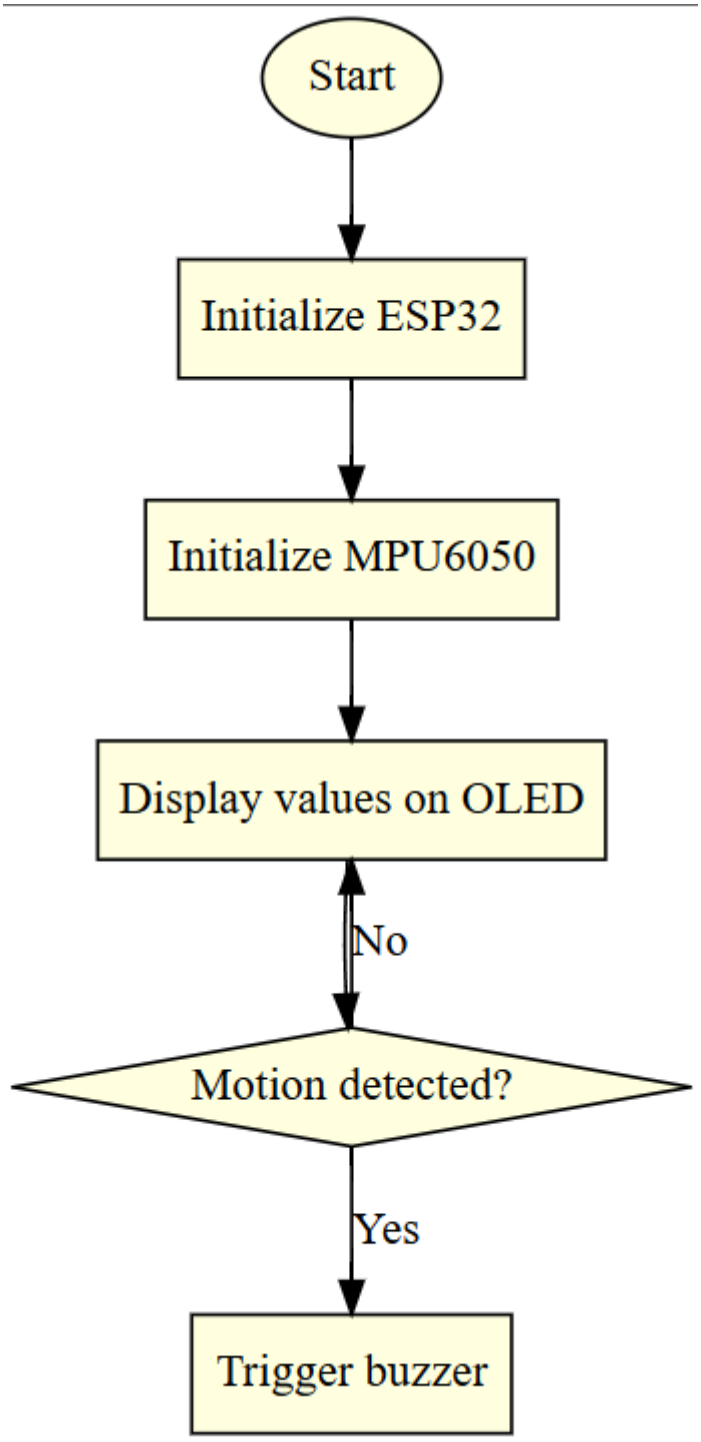
Circuit Connection:



Procedure:

- Connect the ESP32 with the MPU6050 sensor using I²C pins (SDA, SCL, VCC, GND).
- Connect the OLED display to the same I²C pins of the ESP32.
- Connect the buzzer (TMB12A05) to a GPIO pin of the ESP32 and GND.
- Install Arduino IDE and add ESP32 board support.
- Install the required libraries for MPU6050, OLED display, and buzzer control.
- Write or upload the program to the ESP32 using Arduino IDE.
- Power the ESP32 board using a USB cable.
- Observe the X, Y, Z axis values displayed on the OLED screen.
- Tilt the MPU6050 sensor along the Y-axis to test motion detection.
- Check that the buzzer plays a melody when the tilt crosses the threshold.

Flowchart:



Code:

```
#include <Wire.h>
#include <Adafruit_GFX.h>
```

```
#include <Adafruit_SSD1306.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -
1);
Adafruit_MPU6050 mpu;
#define BUZZER_PIN 18   // connect buzzer + to GPIO18, - to GND
void setup() {
  Serial.begin(115200);

  Wire.begin(21, 22); // SDA = 21, SCL = 22 (ESP32 default)

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println("OLED not found!");
    for (;;);
  }
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);

  if (!mpu.begin()) {
    Serial.println("MPU6050 not found!");
    while (1) delay(10);
  }
  Serial.println("MPU6050 ready!");

  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
```

```
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);

  pinMode(BUZZER_PIN, OUTPUT);
}

void loop() {

  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  float x = a.acceleration.x;
  float y = a.acceleration.y;
  float z = a.acceleration.z;



  Serial.print("X: "); Serial.print(x);
  Serial.print(" | Y: "); Serial.print(y);
  Serial.print(" | Z: "); Serial.println(z);

   display.clearDisplay();
  display.setCursor(0, 0);
  display.println("Accel Data (m/s^2):");
  display.print("X: "); display.println(x, 2);
  display.print("Y: "); display.println(y, 2);
  display.print("Z: "); display.println(z, 2);
  display.display();
```

```
  int freq = map(abs(y) * 100, 0, 2000, 200, 2000);

  if (abs(y) > 5) {  // only beep if tilt is noticeable
    tone(BUZZER_PIN, freq, 100); // play tone for 100ms
  } else {
    noTone(BUZZER_PIN); // stop buzzer
  }

  delay(150); // control update speed
}
```

Execution: