RMIT UNIVERSITY

Report – MPU6050

October 2023

Dinh-Son Vu

School of Science, Engineering, and Technology

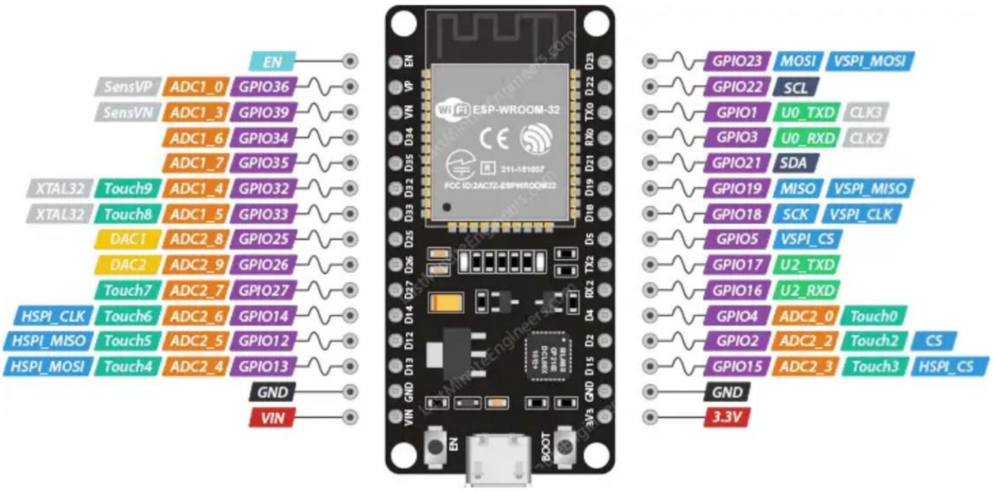
What's next...

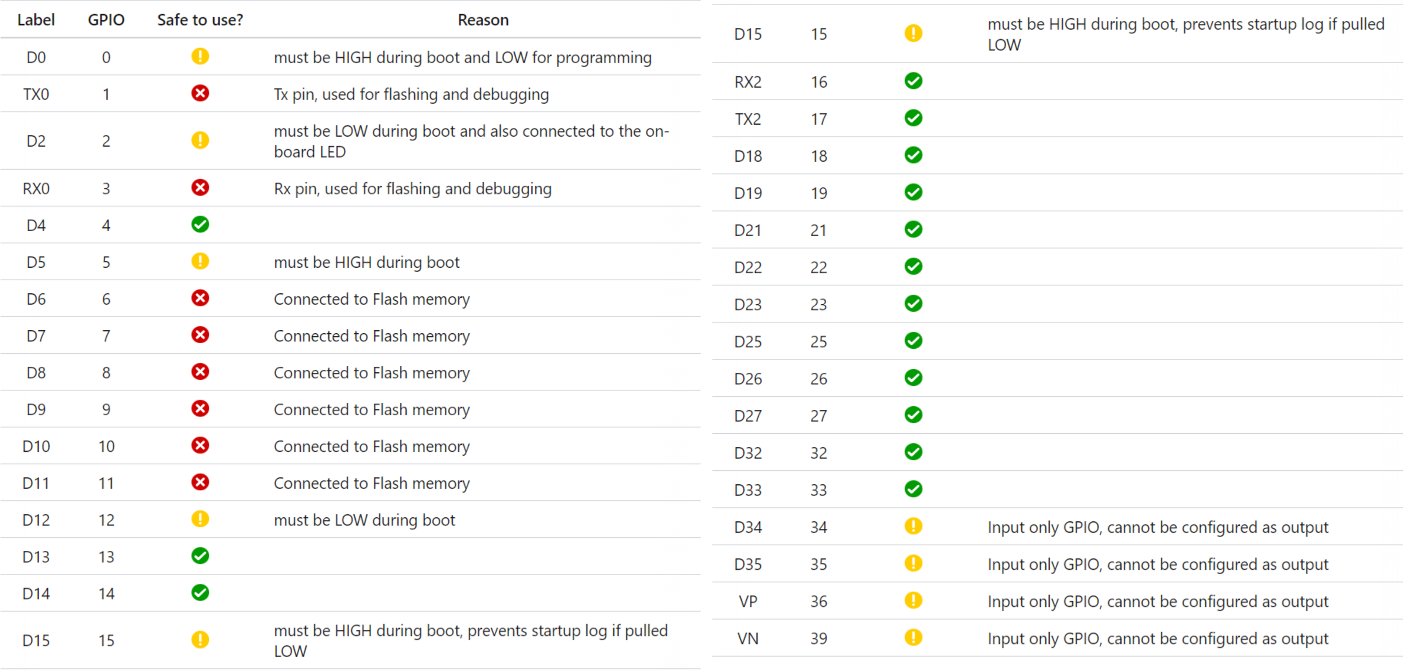
Table of Contents

1. Introduction

This document introduces how to use the inertial measurement unit MPU6050 with an ESP32. This sensor is essential to measure the orientation of the drone.

2. ESP32 Pinout

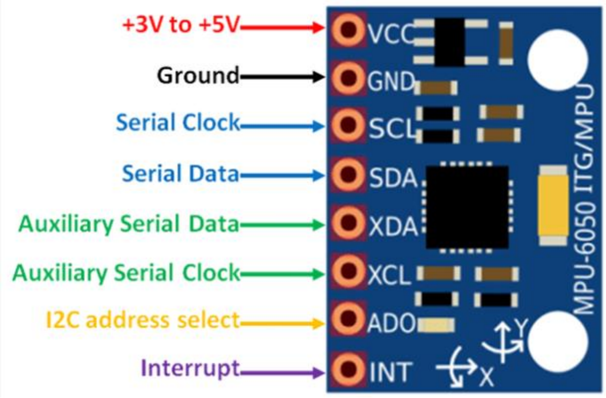




Label	GPIO	Safe to use?	Reason
D0	0	!	must be HIGH during boot and LOW for programming
TX0	1	✗	Tx pin, used for flashing and debugging
D2	2	!	must be LOW during boot and also connected to the on-board LED
RX0	3	✗	Rx pin, used for flashing and debugging
D4	4	✓	
D5	5	!	must be HIGH during boot
D6	6	✗	Connected to Flash memory
D7	7	✗	Connected to Flash memory
D8	8	✗	Connected to Flash memory
D9	9	✗	Connected to Flash memory
D10	10	✗	Connected to Flash memory
D11	11	✗	Connected to Flash memory
D12	12	!	must be LOW during boot
D13	13	✓	
D14	14	✓	
D15	15	!	must be HIGH during boot, prevents startup log if pulled LOW
D15	15	!	must be HIGH during boot, prevents startup log if pulled LOW
RX2	16	✓	
TX2	17	✓	
D18	18	✓	
D19	19	✓	
D21	21	✓	
D22	22	✓	
D23	23	✓	
D25	25	✓	
D26	26	✓	
D27	27	✓	
D32	32	✓	
D33	33	✓	
D34	34	!	Input only GPIO, cannot be configured as output
D35	35	!	Input only GPIO, cannot be configured as output
VP	36	!	Input only GPIO, cannot be configured as output
VN	39	!	Input only GPIO, cannot be configured as output

3. MPU6050 Pinout

The MPU6050 uses a I2C communication. It uses two pins (clock and data) to transmit information. The ESP32 can use any GPIO for the I2C communication, but it uses pin 21 and 22 by default. Note that the MPU6050 has a microchip called a digital motion processor (DMP) that calculates the angles from the gyroscope the accelerometer. However, the DMP only function when the MPU6050 is horizontal with the z-axis pointing upward. Also, note that the I2C communication is highly sensitive to electronic noise: The cables must be a short as possible and away from PWM signal.



MPU6050	ESP32
VCC	3V3
GND	GND
SCL	22
SDA	21

```
#include <Arduino.h>
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#include "Wire.h"

// ================================================================
// Variable declaration
// ================================================================
MPU6050 mpu;       // Prepare the mpu object to obtain the angles from the DMP
MPU6050 accelgyro; // Prepare the accelgyro object to obtain the gyroscope and the
acceleration data

// MPU variable
uint16_t packetSize;    // DMP packet size. Default is 42 bytes.
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer
Quaternion q;           // [w, x, y, z]         quaternion container
VectorFloat gravity;    // [x, y, z]            gravity vector
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll
int16_t ax, ay, az;     // Raw acceleration data from the MPU
int16_t gx, gy, gz;     // Raw gyroscope data from the MPU

float anglex, angley, anglez; // angle in the x, y, z direction
float gyrox, gyroy, gyroz;    // angle rate in the x, y, z direction
float accx, accy, accz;       // acceleration in the x, y, z direction

unsigned long time_prev = 0; // data for the serial communication

// ================================================================
// Function Declaration
// ================================================================
void Init_Serial();     // Function to init the serial monitor
void Init_MPU();        // Function to init the MPU6050
void Get_MPUangle();    // Function to get the angle from the MPU6050
void Get_accelgyro();   // Function to get the gyro and acc from the MPU6050
void SerialDataPrint(); // Function to print data on the serial monitor

// ================================================================
// Setup function
// ================================================================
void setup()
{
  Init_Serial();
  Init_MPU();
}

// ================================================================
// Loop function
// ================================================================
void loop()
{
  Get_MPUangle();
  Get_accelgyro();
  SerialDataPrint();
}

// ================================================================
// Function Definition
// ================================================================
void Init_Serial()
{
  Serial.begin(115200);
  while (!Serial)
    ;
}
// ================================================================
void Init_MPU()
{
  Wire.begin(21, 22);      // Wire.begin(I2C_SDA, I2C_SCL);
  Wire.setClock(400000);   // Set the SCL clock to 400KHz
  accelgyro.initialize();  // Initialize the accelgyro
  mpu.initialize();        // Initialize the MPU
  mpu.dmpInitialize();     // Initialize the DMP (microchip that calculate the
angle on the MPU6050 module)
  mpu.setDMPEnabled(true); // Enable the DMP
  packetSize = mpu.dmpGetFIFOPacketSize();
  mpu.CalibrateAccel(6); // Calibrate the accelerometer
  mpu.CalibrateGyro(6);  // Calibrate the gyroscope
}
// ================================================================
void Get_MPUangle()
{
  // Clear buffer
  mpu.resetFIFO();
  // Get FIFO count
  fifoCount = mpu.getFIFOCount();
  // Wait for the FIFO to be filled with the correct data number
  while (fifoCount < packetSize)
    fifoCount = mpu.getFIFOCount();
  // read a packet from FIFO
  mpu.getFIFOBytes(fifoBuffer, packetSize);
  mpu.dmpGetQuaternion(&q, fifoBuffer);
  mpu.dmpGetGravity(&gravity, &q);
  mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
  anglex = ypr[2] * 180 / M_PI;
  angley = -ypr[1] * 180 / M_PI;
  anglez = -ypr[0] * 180 / M_PI;
}
// ================================================================
void Get_accelgyro()
{
  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
  gyrox = gx / 131.0;
  gyroy = gy / 131.0;
  gyroz = gz / 131.0;
  accx = ax / 16384.;
  accy = ay / 16384.;
  accz = az / 16384.;
}
// ================================================================
void SerialDataPrint()
{
  if (micros() - time_prev >= 20000)
  {
    time_prev = micros();
    Serial.print(millis());
    Serial.print("\t");
    Serial.print(anglex);
    Serial.print("\t");
    Serial.print(angley);
    Serial.print("\t");
    Serial.print(anglez);
    Serial.print("\t");
    Serial.print(gyrox);
    Serial.print("\t");
    Serial.print(gyroy);
    Serial.print("\t");
    Serial.print(gyroz);
    Serial.println();
  }
}
```