Overview

The Pico-10DOF-IMU is an IMU sensor expansion module specialized for Raspberry Pi Pico. It incorporates sensors including a gyroscope, accelerometer, magnetometer, and baroceptor, and uses an I2C bus for communication.
Combined with the Raspberry Pi Pico, it can be used to collect environment sensing data like temperature and barometric pressure or to easily DIY a robot that detects motion gestures and orientations.

Feature

- Standard Raspberry Pi Pico header supports Raspberry Pi Pico series.
- Onboard MPU9250 (3-axis gyroscope, 3-axis accelerometer, and 3-axis magnetometer) for detecting motion gesture, orientation, and magnetic field.
- Onboard LPS22HB barometric pressure sensor, for sensing the atmospheric pressure of the environment.
- Provides online complete manual and resources (example programs such as Raspberry Pi Pico C/C++ and Micro Python).

Specifications

Sensor	Parameters
Acceslerometer	Resolution: 16 bits Measuring range (optional): ±2, ±4, ±8, ±16g Operating current: 450uA
Gyroscope	Resolution: 16 bits Measuring range (optional): ±250, ±500, ±1000, ±2000°/sec Operating current: 3.2mA
Magnetometer	Resolution: 14 bits Measuring range: ±4800μT Operating current: 280uA
Baroceptor	Measuring range: 260 ~ 1260hPa Measuring accuracy (ordinary temperature): ±0.025hPa Measuring speed: 1Hz - 75Hz
Electric	**Parameters**
Operating voltage	5V

Hardware Description

- Pico-10DOF-IMU has three revisions:
  - 1. USB silkscreen update, add XYZ axis silkscreen and add 0R resistor for power management.
  - 2. The FSYNC, ICM.INT, and LPS.INT pins are respectively connected to two groups of GPIOs with 0-ohm resistors to avoid GPIO conflicts when sharing with other modules.
  - 3. Use MPU9250 instead of ICM20948, and change the silkscreen name to Pico-10DOF-IMU Rev2.1.

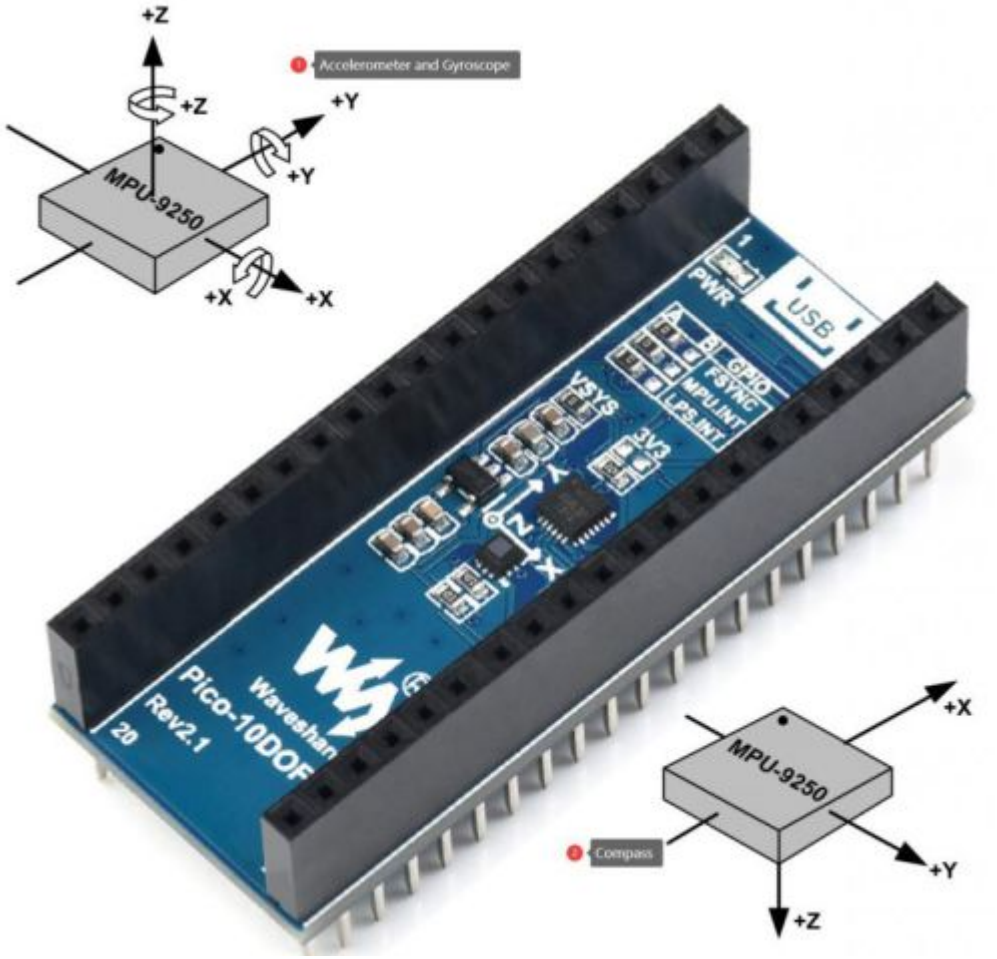
Hardware Connection

1. Note that the USB Logo on the Pico-10DOF-IMU Rev2.1 corresponds to the USB connection direction of the Raspberry Pi Pico.
2. When downloading the C program, be sure to press and hold the BOOT key before connecting the USB cable.

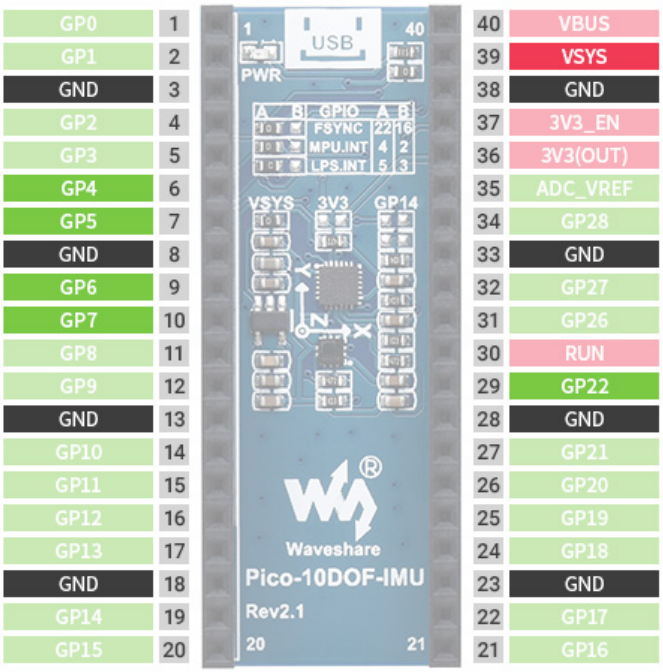
Axial Description

The axes of the accelerometer, gyroscope, and magnetometer on the MPU9250 are shown in the figure below. The magnetometer on the MPU9250 will be interfered with by hard magnetic, so when the data read by the magnetometer is fitted with an

ellipsoid, the sphere is off-center and does not Circle. This will bring an initial magnetic field offset to the magnetometer, making the magnetometer data eccentric. The magnetometer needs to be initialized when the power is turned on. Please refer to the initialization chapter below.



Pinout

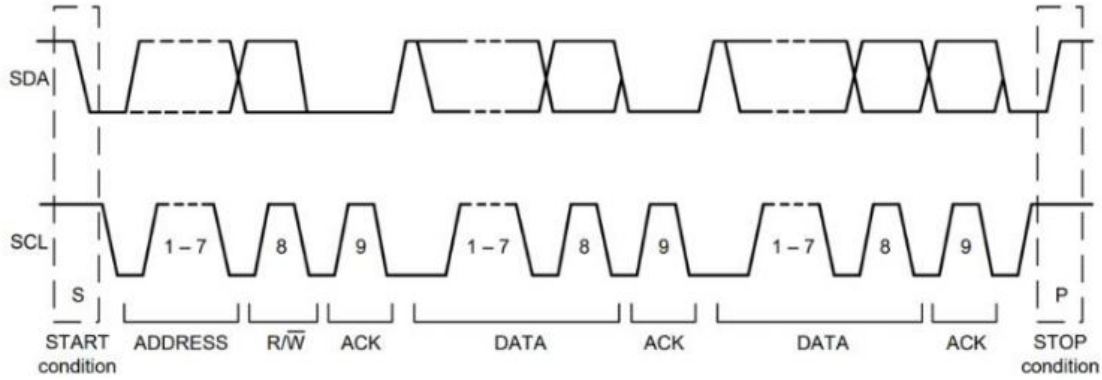


VSYS		5V Power
GND		Ground
GP4	MPU.INT	MPU9250 INT pin
GP5	LPS.INT	LPS22HB INT pin
GP6	SDA	I2C SDA pin
GP7	SCL	I2C SCL pin
GP22	FSYNC	MPU9250 FSYNC pin

1. Pico-10DOF-IMU Rev2.1 uses GPIO as shown in the figure above, in which SDA (GPIO6), SCL (GPIO7) pins are fixedly connected, MPU9250 INT (GPIO4), FSYNC (GPIO22), LPS22HB INT (GPIO5) can be connected through 0R The resistance change connection pins are MPU9250 INT (GPIO22), FSYNC (GPIO16), LPS22HB INT (GPIO3), click to view the schematic diagram for details.
2. The 40Pin of Pico-10DOF-IMU Rev2.1 is powered by the VSYS pin of Raspberry Pi Pico by default. If you want to turn off the 10DOS power supply, you can solder the 0R of R13 to R15 so that you can use the GPIO14 of Raspberry Pi Pico to turn on /off the 10DOF power supply. Please click to view the schematic diagram for details.
3. If you want to use the 3.3V of Pico as the power supply, you can solder the 0R of R13 to R12. Please click to view the schematic diagram for details.
4. If you want to remove the LED of Pico-10DOF-IMU Rev2.1 and the 0R on the R11, you can click to view the schematic diagram for details.

I2C Bus

- The Pico-10DOF-IMU onboard MPU9250 uses the I2C bus for communication. The read and write timing diagram is shown in the figure below. For more details, please refer to datasheet.



Complete I²C Data Transfer

Single-Byte Write Sequence

Master	S	AD+W		RA		DATA		P
Slave			ACK		ACK		ACK	

Burst Write Sequence

Master	S	AD+W		RA		DATA		DATA		P
Slave			ACK		ACK		ACK		ACK	

Single-Byte Read Sequence

Master	S	AD+W		RA		S	AD+R			NACK	P
Slave			ACK		ACK			ACK	DATA		

Burst Read Sequence

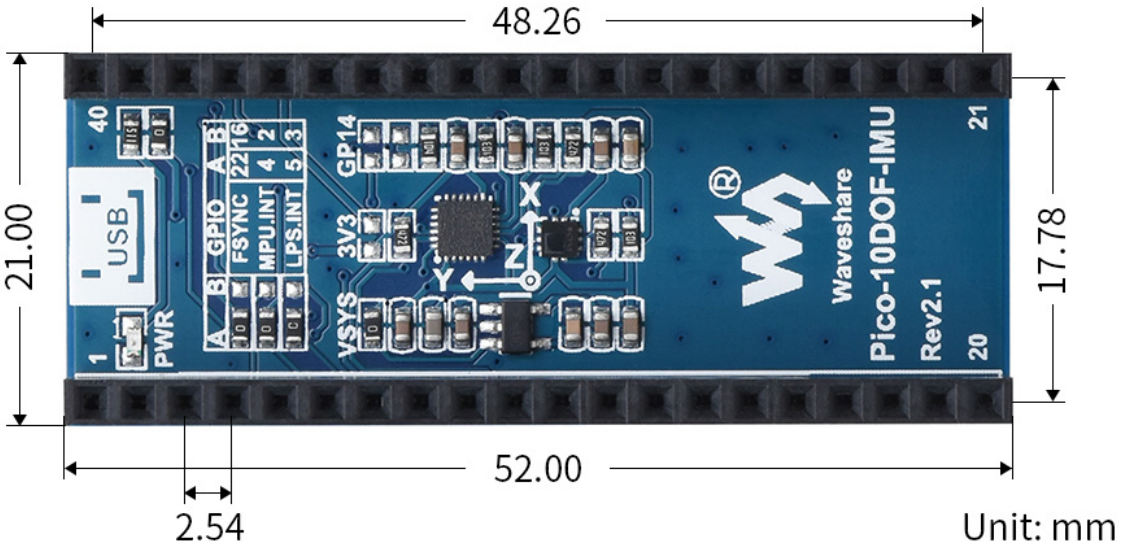
Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

Signal	Description
S	Start Condition: SDA goes from high to low while SCL is high
AD	Slave I²C address
W	Write bit (0)
R	Read bit (1)
ACK	Acknowledge: SDA line is low while the SCL line is high at the 9th clock cycle
NACK	Not-Acknowledge: SDA line stays high at the 9th clock cycle
RA	MPU-9250 internal register address
DATA	Transmit or received data
P	Stop condition: SDA going from low to high while SCL is high

1. Raspberry Pi Pico, as the Master device, pulls down SDA successively, and the SCL pin initiates the START condition of the I2C bus, and then writes the device address (7bits) and write command (1bit) with a total of 8 bits of data. If the pin is connected correctly, 10ODF is used as the slave device. Send an ACK response.

2. Raspberry Pi Pico continues to write the register address (RA) and register value (DATA) respectively and waits for the ACK response. After writing, the Raspberry Pi Pico pulls up SCL successively, and the SDA pin sends a STOP condition.
3. If the Raspberry Pi Pico reads the DATA of the register (RA), when writing RA and waiting for the ACK response, it re-initiates the START condition, and then writes the device address (7bits) and the read command (1bit) for a total of 8bits and waits for the ACK response. 10DOF returns to DATA. After Pico receives DATA, keep SDA high.
4. Please refer to the burst Read/Write Sequence in the figure above for the continuous write register value.

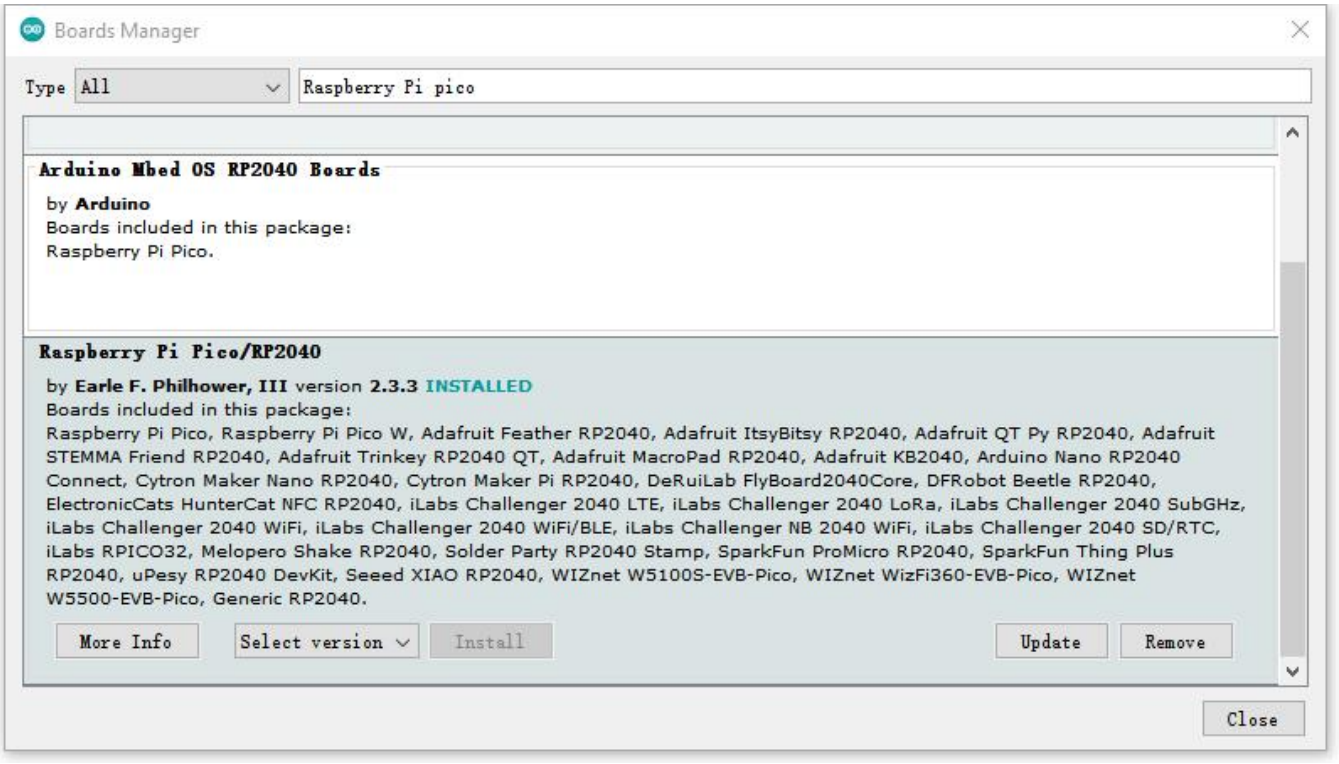
Outline Dimensions



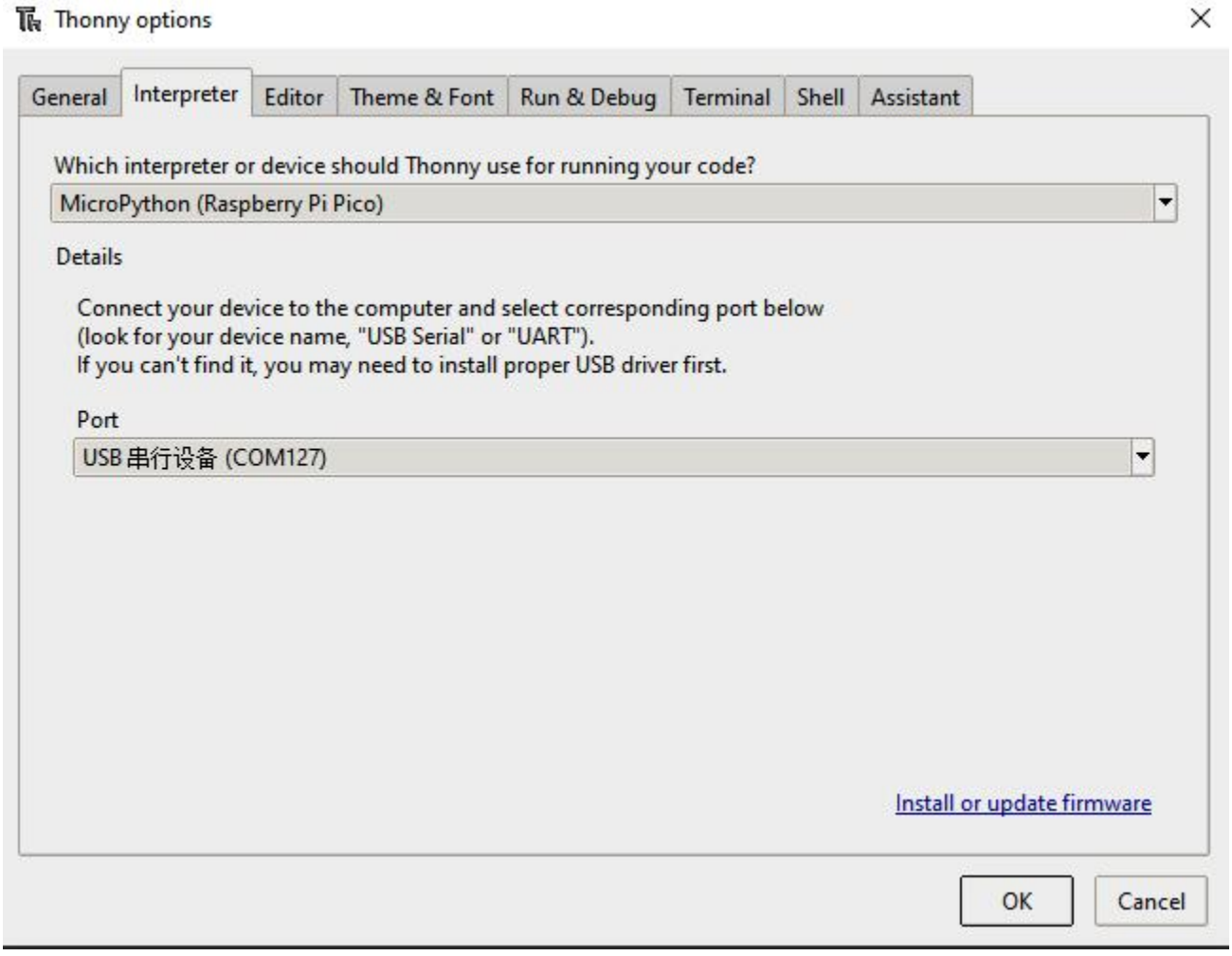
Environment Building

- We test the code with Arduino IDE and Thony, click to download the related IDE, and open them after installation.

1. Install Pico SDK on Arduino IDE, click Tools->Board->Boards Manager, then search "Raspberry Pi Pico", and find the corresponding libraries to install.

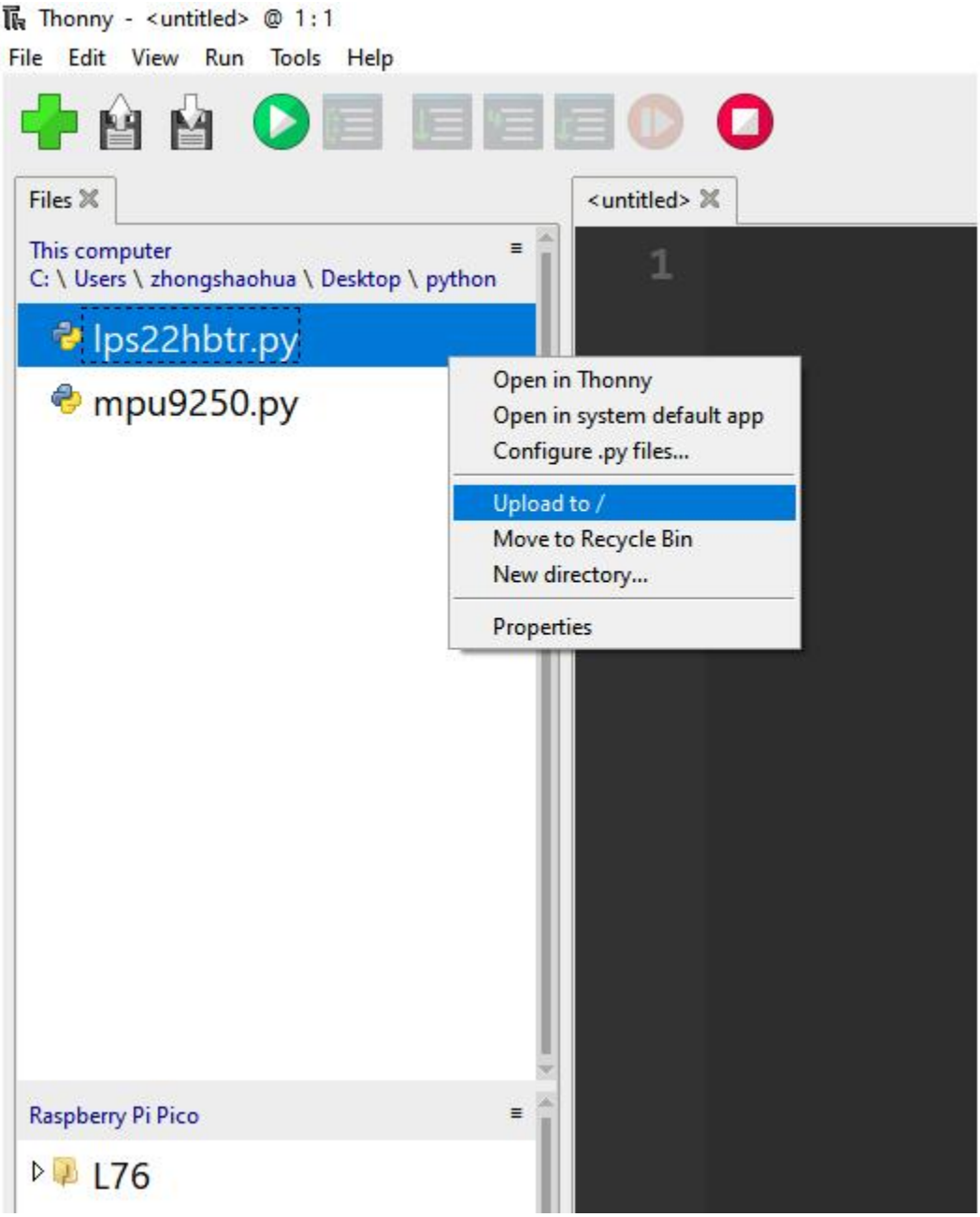


2. Please refer to Micropython official document and set up python environment, select the Raspberry Pi Pico device in Thonny's Tools->Options->Interprete, as shown as below.



Demo Download

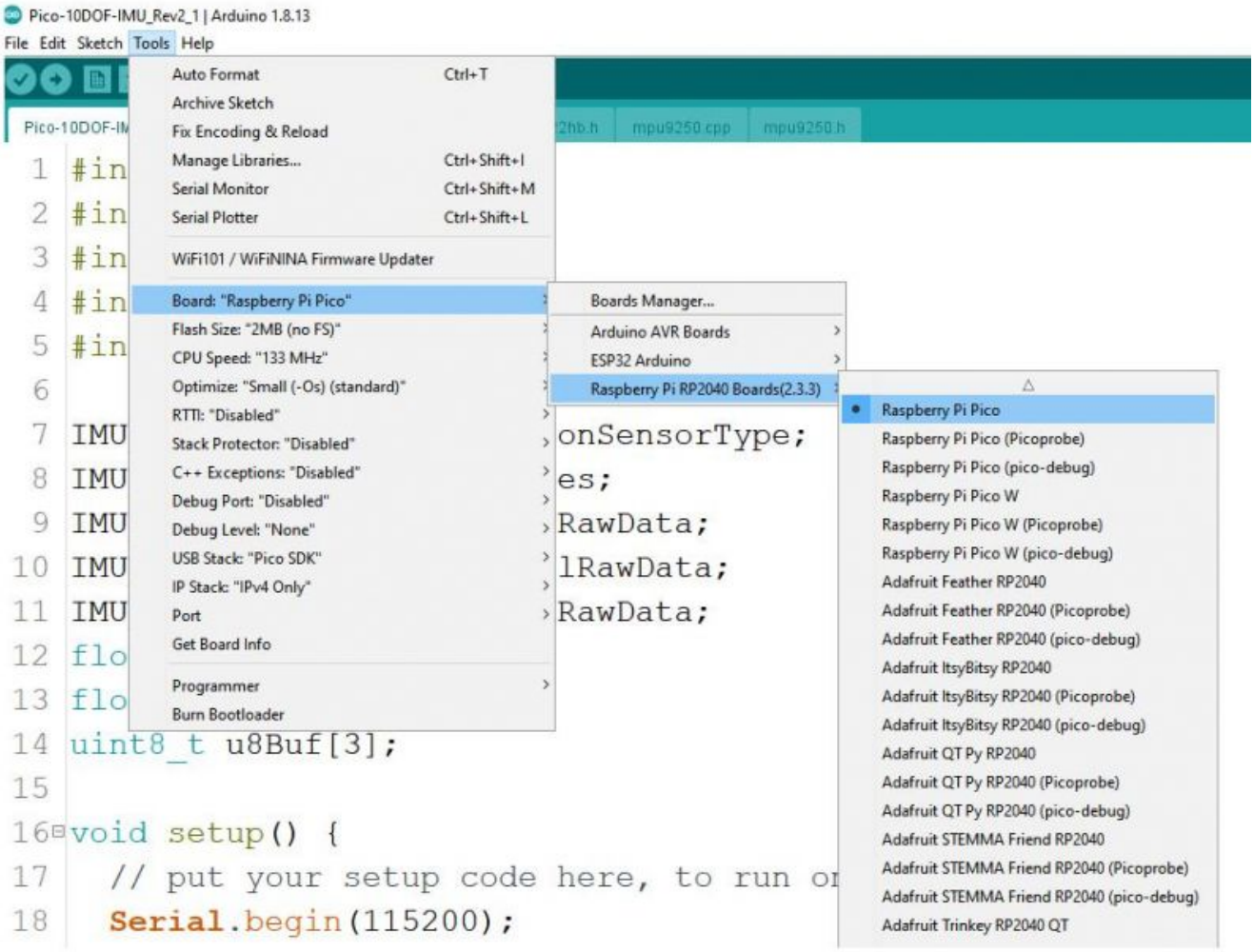
1. Click to download sample demo.
2. Unzip the sample demo, click .ino directly to open the Arduino sample demo, and upload the Micorpython sample demo to the Pico file system, as shown in the figure.



Demo Usage

Arduino

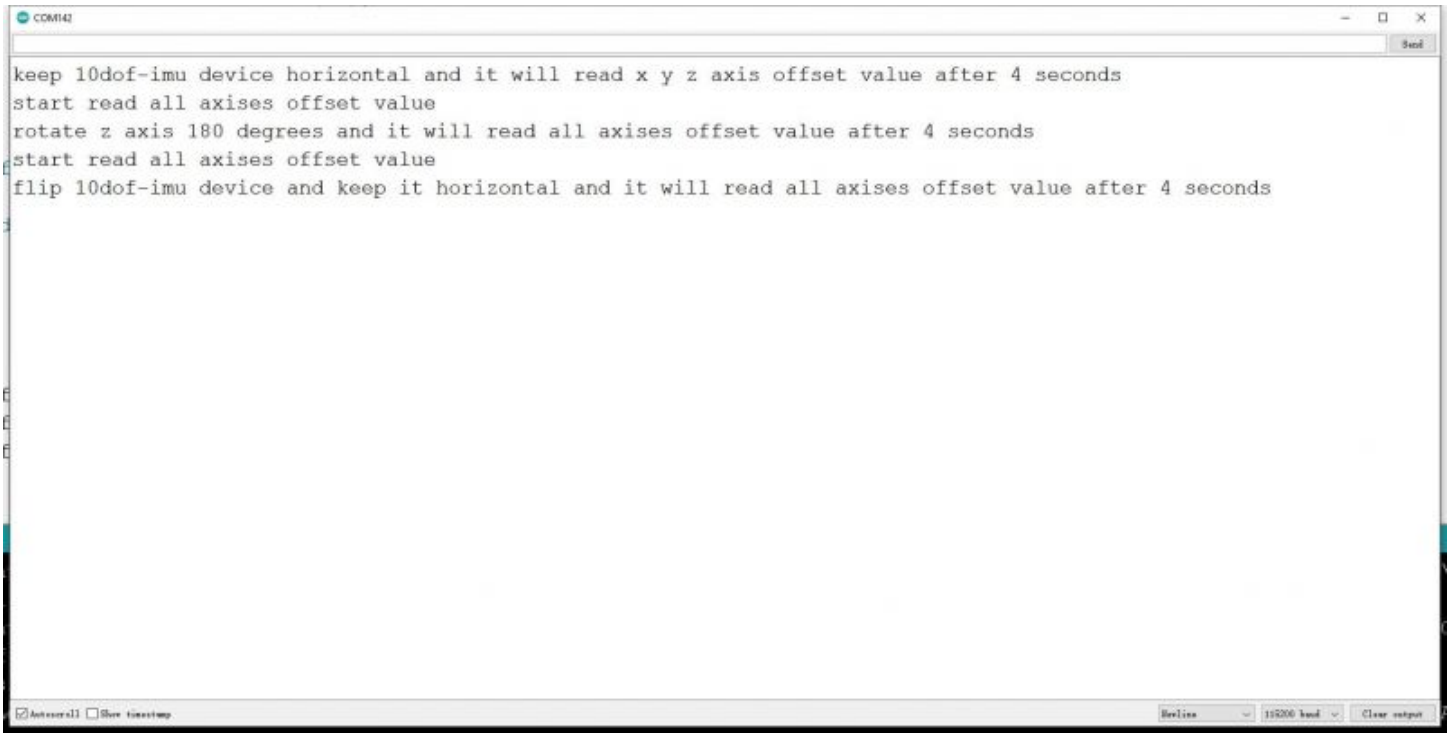
1. Press and hold the BOOT button on the Pico and then connect the USB cable, open the .ino sample demo, click the menu bar and select Tools->Board->Raspberry Pi Pico. As shown in the figure below.



2. Click upload under Edit to download the code to Pico. After downloading, open the device manager to view the virtual COM number of Pico, and then select the corresponding COM number in Tools->Ports.

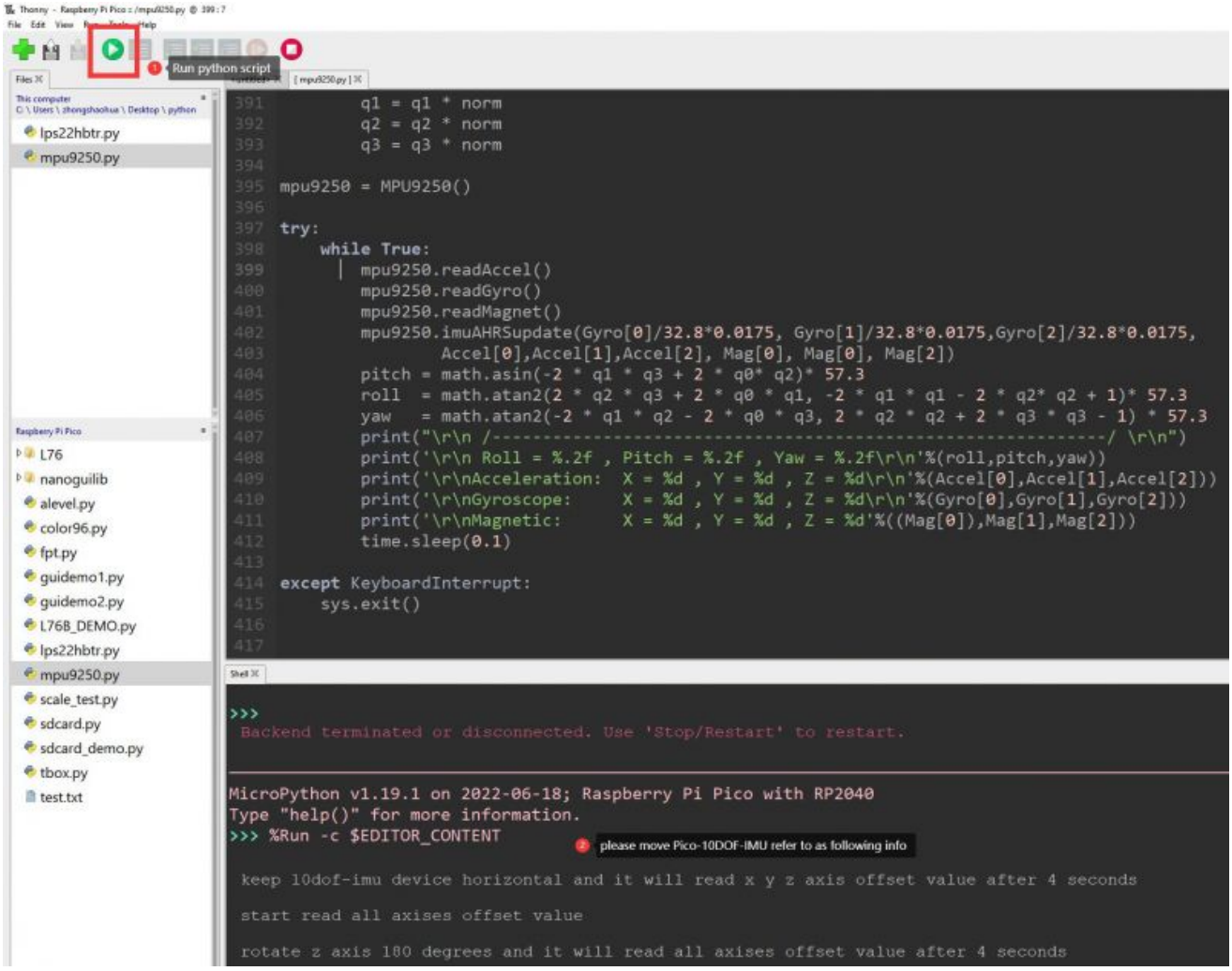
3. Click to open Monitor Serial at the top right of the Arduino IDE, and follow the serial port prompts to initialize the Pico-10DOF-IMU. For details, please refer to the Pico-

10DOF-IMU initialization chapter.



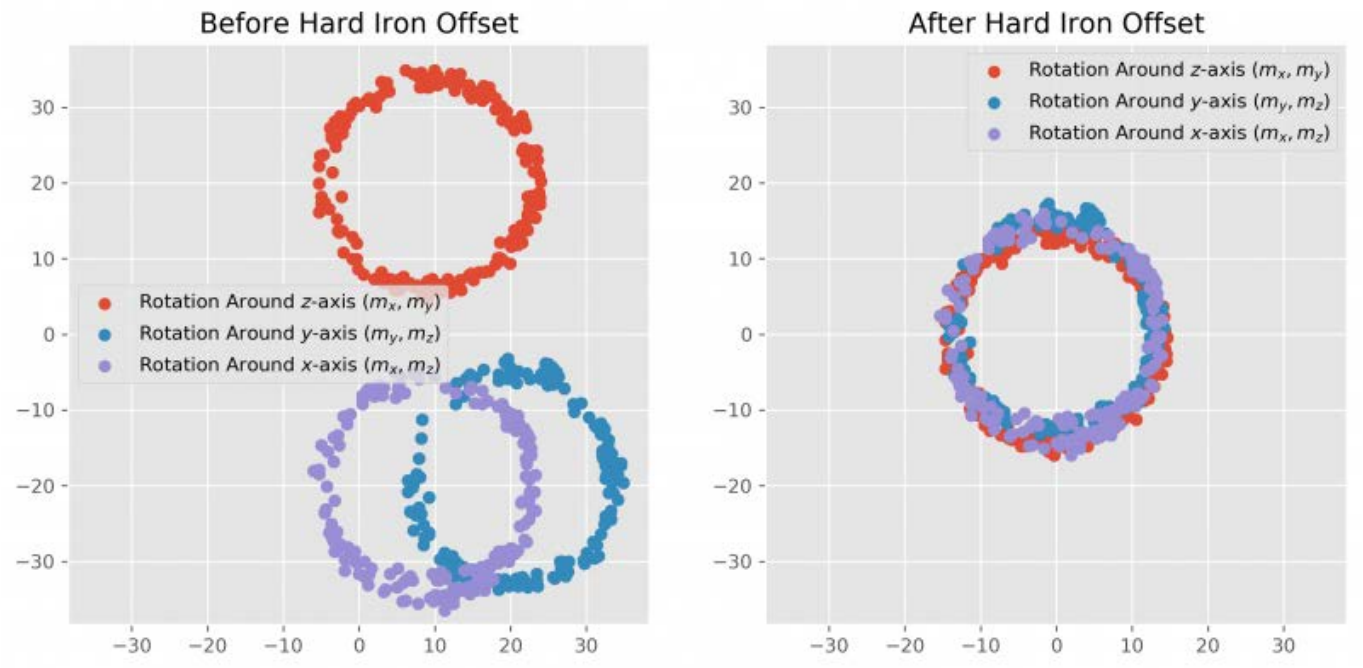
Micropython

- Open the mpu9250.py, lps22hb.py scripts in the Pico file system, and click Run to run, where mpu9250.py will output relevant information to initialize the configuration of Pico-10DOF-IMU, as shown in the figure below, please refer to Pico-10DOF for the detailed process -IMU initialization chapter.



Initialization

- The magnetometer on the MPU9250 will be interfered with by the hard magnetic field. When the ellipsoid fitting is performed on the data read by the magnetometer, the sphere is off-center and not round. This will bring an initial magnetic field offset to the magnetometer, making the magnetometer Data eccentricity, as shown in the figure below:



- After power-on, initialize according to the prompt information sent by the serial port. Calculate the eccentric offset value of the magnetometer, as shown in the following figure:

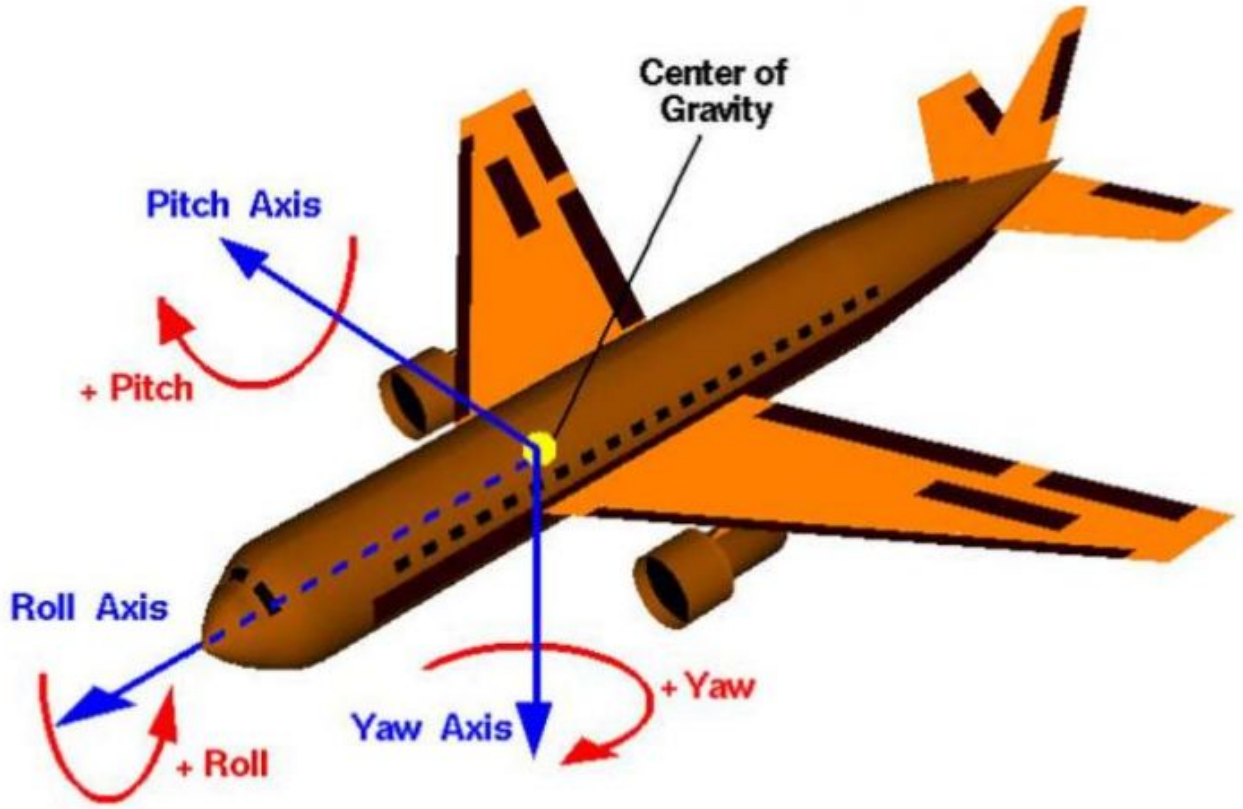

```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

1  keep 10dof-imu device horizontal and it will read x y z axis offset value after 4 seconds

   start read all axises offset value

2  rotate z axis 180 degrees and it will read all axises offset value after 4 seconds

   start read all axises offset value

3  flip 10dof-imu device and keep it horizontal and it will read all axises offset value after 4 seconds

   start read all axises offset value
```

Demo Analysis

This section briefly analyzes the mpu9250.py sample program.

- mpu9250 = MPU9250() instantiates the MPU9250 class. The instantiation process will include gyroscope initialization and geomagnetic calibration.

- mpu9250.readAccel() , mpu9250.readGyro() , mpu9250.readMagnet() respectively read raw data such as accelerometer, gyroscope, geomagnetometer, etc.
- mpu9250.imuAHRSupdate() is the mahony algorithm used to convert the values of the accelerometer, gyroscope and geomagnetometer into Euler angles (pitch, roll, yaw). Please click the link to view the detailed process of the mahony algorithm
- Euler angles are shown in the figure below:




```
mpu9250 = MPU9250()


try:
    while True:
        mpu9250.readAccel()
        mpu9250.readGyro()
        mpu9250.readMagnet()
        mpu9250.imuAHRSupdate(Gyro[0]/32.8*0.0175, Gyro[1]/32.8*0.0175,Gyro[2]/3
2.8*0.0175,
                  Accel[0],Accel[1],Accel[2], Mag[0], Mag[0], Mag[2])
        pitch = math.asin(-2 * q1 * q3 + 2 * q0* q2)* 57.3
        roll  = math.atan2(2 * q2 * q3 + 2 * q0 * q1, -2 * q1 * q1 - 2 * q2* q2
+ 1)* 57.3
```

```
        yaw   = math.atan2(-2 * q1 * q2 - 2 * q0 * q3, 2 * q2 * q2 + 2 * q3 * q3
- 1) * 57.3

        print("\r\n /-------------------------------------------------------------
--/ \r\n")

        print('\r\n Roll = %.2f , Pitch = %.2f , Yaw = %.2f\r\n'%(roll,pitch,yaw
))

        print('\r\nAcceleration:  X = %d , Y = %d , Z = %d\r\n'%(Accel[0],Accel[
1],Accel[2]))

        print('\r\nGyroscope:     X = %d , Y = %d , Z = %d\r\n'%(Gyro[0],Gyro[1]
,Gyro[2]))

        print('\r\nMagnetic:      X = %d , Y = %d , Z = %d'%((Mag[0]),Mag[1],Mag
[2]))

        time.sleep(0.1)


except KeyboardInterrupt:

    sys.exit()
```

Resource

Documents

- Schematic Disgram
- ICM20948 Datasheet
- LPS22HB Specification
- LSF0204d Specification
- MPU9250

Demo Codes

- Demo code

Software

- Thonny-3.3.3.zip

- Zimo221.7z
- Image2Lcd

Download Firmware