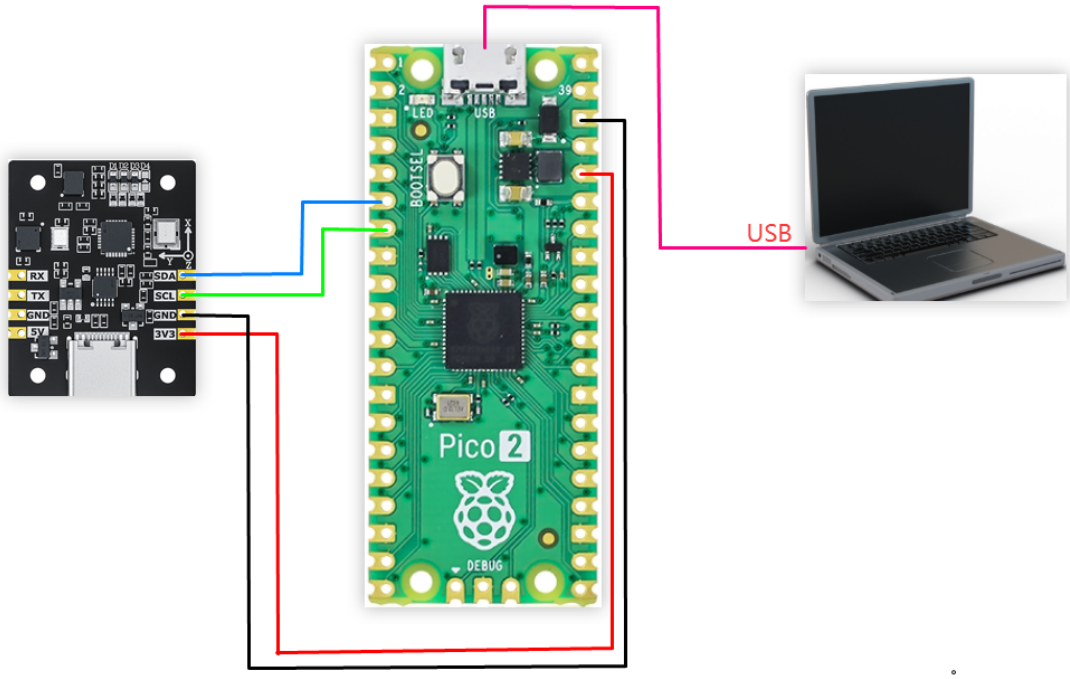
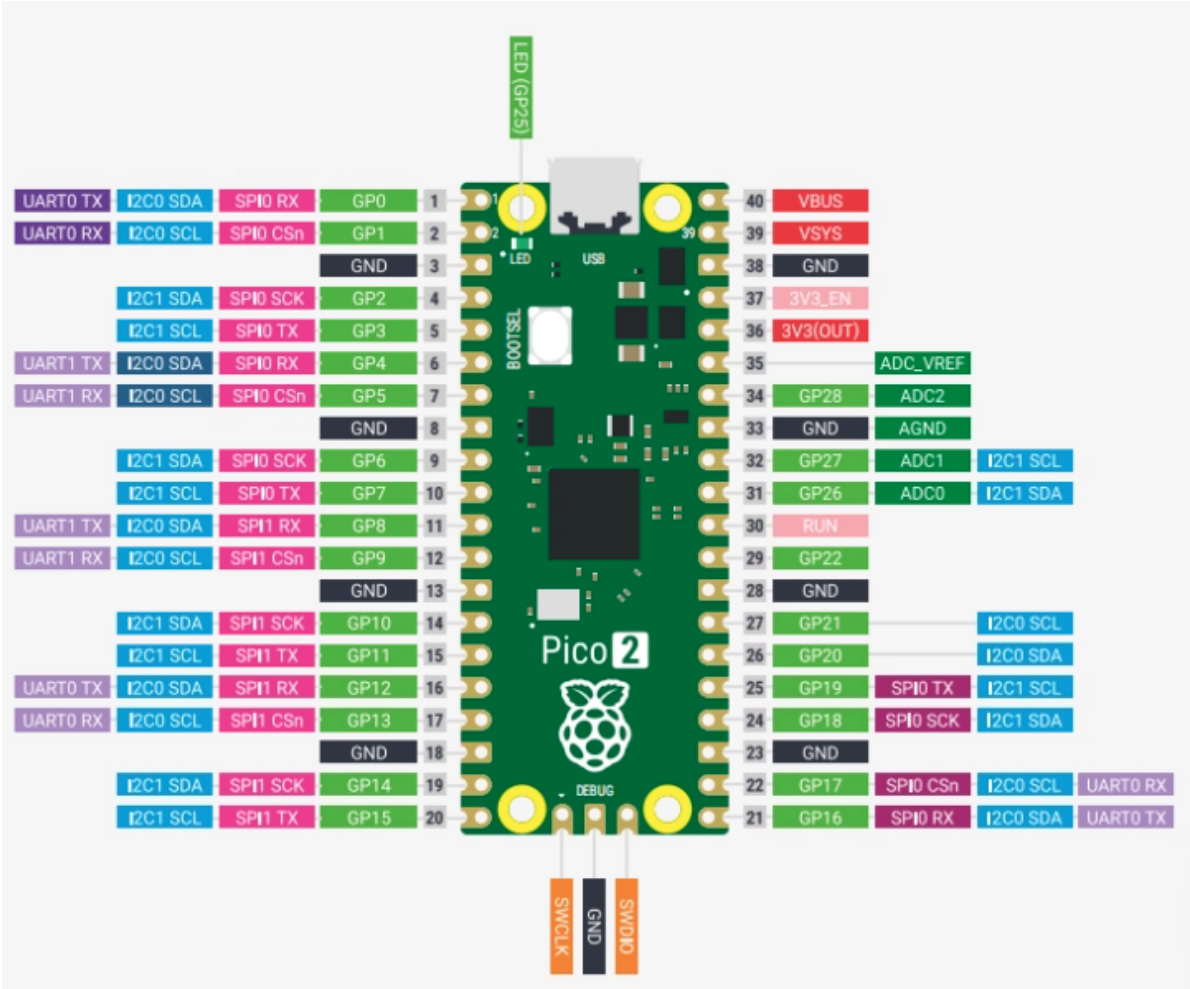
Pico-IIC Data Reading

This example uses the Pico2 development board, a Windows computer, several DuPont wires, and an IMU attitude sensor.

1. Connecting the Device



IMU Attitude Sensor	Pico2
SDA	GP4
SCL	GP5
GND	GND
5V	3V3(OUT)



2. Key Code Analysis

Please refer to the source code in the documentation for specific code.

```
# Get accelerometer triaxial data, return accel=[a_x, a_y, a_z]
    def get_accelerometer_data(self):
        values = self._read_data(self.FUNC_RAW_ACCEL, 6)
        # 转化单位为g
        accel_ratio = 16 / 32767.0
        a_x = struct.unpack('h', bytearray(values[0:2]))[0]*accel_ratio
        a_y = struct.unpack('h', bytearray(values[2:4]))[0]*accel_ratio
        a_z = struct.unpack('h', bytearray(values[4:6]))[0]*accel_ratio
        accel = [a_x, a_y, a_z]
        if self._delay_time > 0:
            time.sleep(self._delay_time)
        return accel


    # Get the gyro triaxial data, return gyro=[g_x, g_y, g_z]
    def get_gyroscope_data(self):
        values = self._read_data(self.FUNC_RAW_GYRO, 6)
        # 转化单位为rad/s
        AtoR = math.pi / 180.0
        gyro_ratio = (2000 / 32767.0) * AtoR
        g_x = struct.unpack('h', bytearray(values[0:2]))[0]*gyro_ratio
        g_y = struct.unpack('h', bytearray(values[2:4]))[0]*gyro_ratio
        g_z = struct.unpack('h', bytearray(values[4:6]))[0]*gyro_ratio
        gyro = [g_x, g_y, g_z]
        if self._delay_time > 0:
```

```
            time.sleep(self._delay_time)
        return gyro


    # Get the quaternion of the IMU, return quat=[w, x, y, z]
    def get_imu_quaternion_data(self):
        values = self._read_data(self.FUNC_QUAT, 16)
        q0 = struct.unpack('f', bytearray(values[0:4]))[0]
        q1 = struct.unpack('f', bytearray(values[4:8]))[0]
        q2 = struct.unpack('f', bytearray(values[8:12]))[0]
        q3 = struct.unpack('f', bytearray(values[12:16]))[0]
        quat = [q0, q1, q2, q3]
        if self._delay_time > 0:
            time.sleep(self._delay_time)
        return quat

  # Get the board's attitude angle, returning euler=[roll, pitch, yaw]
# ToAngle=True returns the angle, ToAngle=False returns the radians。
    def get_imu_attitude_data(self, ToAngle=True):
        values = self._read_data(self.FUNC_EULER, 12)
        roll = struct.unpack('f', bytearray(values[0:4]))[0]
        pitch = struct.unpack('f', bytearray(values[4:8]))[0]
        yaw = struct.unpack('f', bytearray(values[8:12]))[0]
        if ToAngle:
            RtoA = 180.0 / math.pi
            roll = roll * RtoA
            pitch = pitch * RtoA
            yaw = yaw * RtoA
        euler = [roll, pitch, yaw]
        if self._delay_time > 0:
            time.sleep(self._delay_time)
        return euler
```

get_accelerometer_data(): Gets the accelerometer's three-axis data, returning accel=[a_x, a_y, a_z]

get_gyroscope_data(): Gets the gyroscope's three-axis data, returning gyro=[g_x, g_y, g_z]

get_imu_quaternion_data(): Gets the IMU's quaternion, returning quat=[w, x, y, z]

get_imu_attitude_data(): Gets the board's attitude angle, returning euler=[roll, pitch, yaw]

3. Reading IMU Data

After the program runs, the IMU data can be displayed in the terminal below.

IIC.py

```python
from machine import I2C, Pin
import struct
import time
import math


READ_INTERVAL = 0.1  # 传感器读取间隔（秒） Sensor read interval in seconds


class YbImuI2c(object):

    def __init__(self, debug=False):

        # 创建I2C通信对象   Create I2C communication object
        self.dev = I2C(0, scl=Pin(5), sda=Pin(4), freq=400000)  # 使用I2C0，SCL=GP5，SDA=GP4
                                                                # Use I2C0, SCL=GP5, SDA=GP4

        self._delay_time = 0.001
        self._debug = debug
        self._addr = 0x23

        self.FUNC_VERSION = 0x01
```

Shell

```
------ Sensor Data ------
Acceleration [g]:       x=-0.914, y=-1.100, z= 0.721
Gyroscope [rad/s]:      x=-0.837, y= 1.670, z=-0.289
Magnetometer [uT]:      x=-43.751, y= 5.591, z=-30.348
Quaternion:             w=-0.07147, x= 0.36896, y= 0.35920, z=-0.85227
Euler Angle [deg]:      roll=-54.77, pitch= 35.28, yaw= 151.44
Barometer:              height= 0.25 m, temperature= 41.95 °C
                        pressure= 101479.05469 Pa, pressure_diff= 101482.06250 Pa
--------------------------
```

MicroPython (Raspberry Pi Pico)

Note: The above is for reading data from a 10-axis IMU; 6-axis data is without a magnetometer and barometer; 9-axis data is without a barometer.