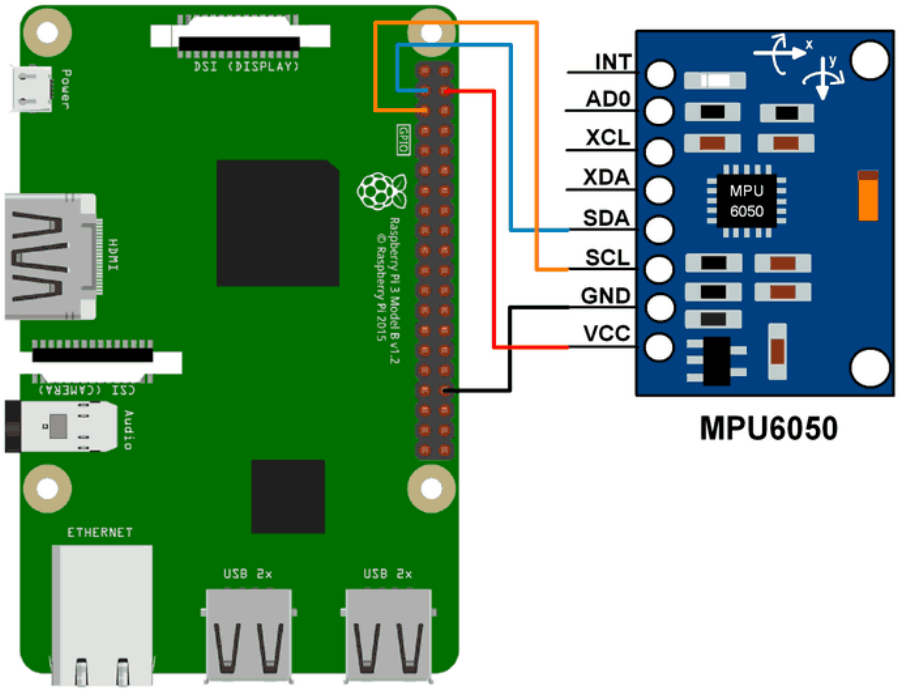
3. MPU6050 Accelerometer (I2C)

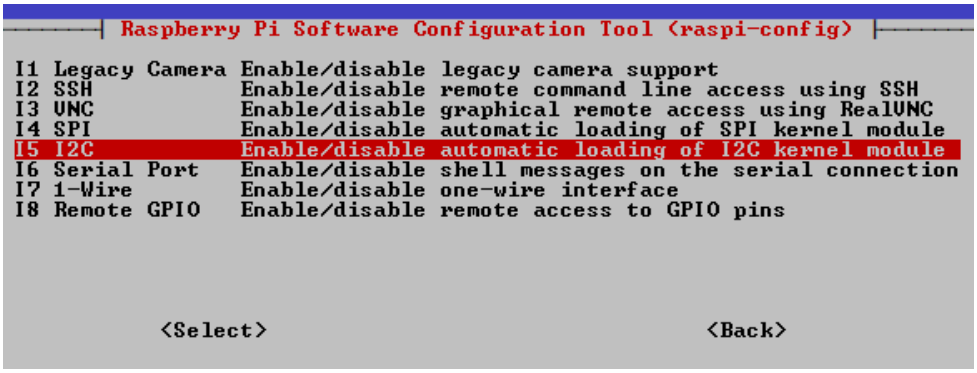


The MPU6050 sensor module is a Micro Electro-Mechanical Systems (MEMS) that consists of a 3-axis Accelerometer and 3-axis Gyroscope. It is a complete 6-axis motion tracking device that combines a 3-axis gyroscope, 3-axis accelerometer, and digital motion processor in one small package. The MPU6050 sensor module helps to measure acceleration, velocity, orientation, displacement, and many other motion-related parameters of a system or object.

Gyro Range: ± 250, 500, 1000, 2000 °/s (degrees per second)

Accelerometer Range: ± 2 ± 4 ± 8 ± 16 g

a. Connect the **SDA**, **SCL**, **VCC** and **GND** lines to your Pi 3 board as shown in the diagram above. Note: When you leave the ADO pin unconnected, the default I2C address is **0x68** HEX and when you connect it to 3.3V, the I2C address becomes **0x69** HEX.
b. Configure the operating system support for I2C via **sudo respi-config** or the configuration GUI.



c. After that, you need to install the **smbus** package via step 1a or step 1b:

1a. install the python-smbus package

sudo apt install python3-smbus

1b. Install this package from PyPi repository

pip install mpu6050-raspberrypi

d. If everything is configured correctly, you should be able to write a python program that interacts with the MPU6050 sensor. Below is just one Python example for MPU6050.

```
import smbus2

import time

# MPU-6050 Registers and Address

MPU6050_ADDR = 0x68

PWR_MGMT_1 = 0x6B

ACCEL_XOUT_H = 0x3B

GYRO_XOUT_H = 0x43

# Initialize I2C bus

bus = smbus2.SMBus(1)

# Wake up the MPU-6050

bus.write_byte_data(MPU6050_ADDR, PWR_MGMT_1, 0)

def read_raw_data(addr):

  # Read two bytes of data from the given address

  high = bus.read_byte_data(MPU6050_ADDR, addr)

  low = bus.read_byte_data(MPU6050_ADDR, addr+1)

  value = (high << 8) | low

  # Convert to signed value

  if value > 32767:
```

```
        value -= 65536
    return value
try:
    while True:
        # Read accelerometer data
        accel_x = read_raw_data(ACCEL_XOUT_H)
        accel_y = read_raw_data(ACCEL_XOUT_H + 2)
        accel_z = read_raw_data(ACCEL_XOUT_H + 4)
        # Read gyroscope data
        gyro_x = read_raw_data(GYRO_XOUT_H)
        gyro_y = read_raw_data(GYRO_XOUT_H + 2)
        gyro_z = read_raw_data(GYRO_XOUT_H + 4)
        # Convert raw data to meaningful values (optional scaling may be needed)
        accel_x_scaled = accel_x / 16384.0
        accel_y_scaled = accel_y / 16384.0
        accel_z_scaled = accel_z / 16384.0
        gyro_x_scaled = gyro_x / 131.0
        gyro_y_scaled = gyro_y / 131.0
        gyro_z_scaled = gyro_z / 131.0
        print(f"Accelerometer: X={accel_x_scaled:.2f}, Y={accel_y_scaled:.2f}, Z={accel_z_scaled:.2f}")
        print(f"Gyroscope: X={gyro_x_scaled:.2f}, Y={gyro_y_scaled:.2f}, Z={gyro_z_scaled:.2f}")
        time.sleep(1)
except KeyboardInterrupt:
    print("Exiting...")
```

Reference:

1. A Step-by-Step Guide to Reading MPU6050 Data with Raspberry Pi and Python - Machine Learning Site
2. Using the MPU-6050 with the Raspberry Pi – 38-3D
3. MPU6050 (Accelerometer+Gyroscope) Interfacing with Raspberry Pi |...
4. MPU6050 Gyro Sensor Interfacing with Raspberry Pi