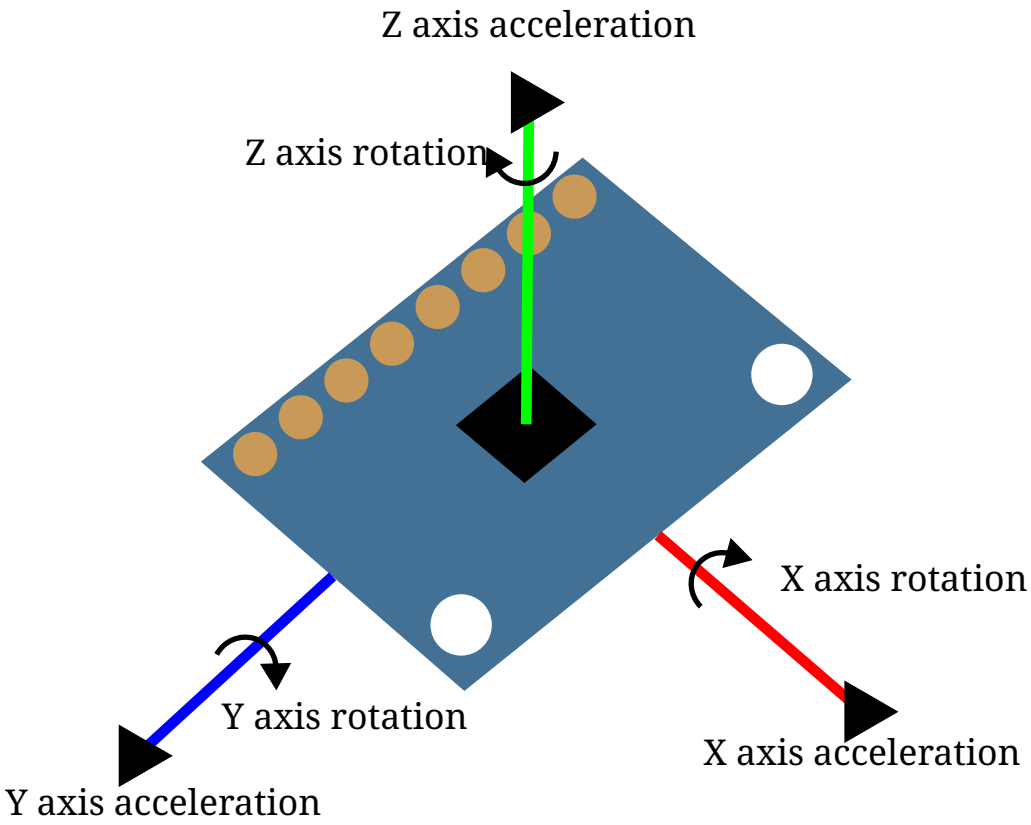
MPU6050 with Raspberry Pi

About MPU6050

The MPU6050 consists of a 3-axis gyroscope and 3-axis accelerometer. It also contains a DMP (digital motion processor) to perform complex calculations, which frees up the controller to do other things.

The MPU6050 has a 16-bit ADC (analog to digital) chip. Because of this, it can receive motion in all three planes.



The MPU6050 uses the I2C interface.

MPU6050 Pinout

Pin	Description
V_{CC}	3-5 V supply voltage
GND	Ground connection
SCL	I2C clock connection
SDA	I2C data connection
XDA	See below
XCL	

Pin	Description
AD0	Address pin
INT	Interrupt pin

XDA and XCL stand for auxiliary data and auxiliary clock, respectively. These pins can be used for interfacing with other I2C devices.

The address pin can be used to change the address of the MPU6050. (default is 0x68)

The interrupt pin is used to indicate that data is available for the microcontroller to read.

MPU6050 Registers

Address	Name in code	Purpose
0x6B	PWR_MGMT_1	Writing to power management register
0x19	SMPLRT_DIV	Writing to sample rate register
0x1A	CONFIG	Writing to configuration register
0x1B	GYRO_CONFIG	Writing to gyro configuration register
0x38	INT_ENABLE	Writing to interrupt enable register
0x3B	ACCEL_X	Reading X Accelerometer raw data
0x3D	ACCEL_Y	Reading Y Accelerometer raw data
0x3F	ACCEL_Z	Reading Z Accelerometer raw data
0x43	GYRO_X	Reading X Gyro raw data
0x45	GYRO_Y	Reading Y Gyro raw data
0x47	GYRO_Z	Reading Z Gyro raw data

Connections to Raspberry Pi

MPU6050 pin	Connects to RPi pin
V_{CC}	3.3 V power
GND	GND
SCL	GPIO 3 (SCL)

MPU6050 pin	Connects to RPi pin
SDA	GPIO 2 (SDA)

Code

I2C needs to be enabled on your Raspberry Pi before running this code. See ADS1115 with Raspberry Pi for details.

This code prints the acceleration and gyro values on each of the three axes.

```
from time import time, sleep
import smbus

bus = smbus.SMBus(1)
previous, current, elapsed = 0, 0, 0
x, y, z = 0, 0, 0

# MPU6050 registers
PWR_MGMT_1  = 0x6B
SMPLRT_DIV  = 0x19
CONFIG      = 0x1A
GYRO_CONFIG = 0x1B
INT_ENABLE  = 0x38
ACCEL_X     = 0x3B
ACCEL_Y     = 0x3D
ACCEL_Z     = 0x3F
GYRO_X      = 0x43
GYRO_Y      = 0x45
GYRO_Z      = 0x47

device_address = 0x68

def init():
    # Write to sample rate register
    bus.write_byte_data(device_address, SMPLRT_DIV, 7)
    # Write to power management register
    bus.write_byte_data(device_address, PWR_MGMT_1, 1)
    # Write to Configuration register
    bus.write_byte_data(device_address, CONFIG, 0)
    # Write to Gyro configuration register
    bus.write_byte_data(device_address, GYRO_CONFIG, 24)
    # Write to interrupt enable register
    bus.write_byte_data(device_address, INT_ENABLE, 1)

def read_data(addr):
    # Get elapsed time for calculating gyro angle
    global current
    previous = current
    current = time()
```

```
    elapsed = current - previous

    # Accel and Gyro value are 16-bit
    high = bus.read_byte_data(device_address, addr)
    low = bus.read_byte_data(device_address, addr + 1)

    # Concatenate higher and lower value
    value = (high << 8) | low

    # Get signed value from sensor
    if value > 32768: value -= 65536
    return value

def scale_gyro(val):
    return val / 131.0

def scale_accel(val):
    return val / 16384.0

init()
while True:
    try:
        x_accel = scale_accel(read_data(ACCEL_X))
        y_accel = scale_accel(read_data(ACCEL_Y))
        z_accel = scale_accel(read_data(ACCEL_Z))

        x_gyro = scale_accel(read_data(GYRO_X))
        y_gyro = scale_accel(read_data(GYRO_Y))
        z_gyro = scale_accel(read_data(GYRO_Z))

        print("Accel X: %.3f Accel Y: %.3f Accel Z: %.3f Gyro X: %.3f Gyro Y: %.3f
Gyro Z: %.3f" %
              (x_accel, y_accel, z_accel, x_gyro, y_gyro, z_gyro))
        sleep(0.2)
    except KeyboardInterrupt:
        print("Stopped")
        break
```

Output

```
Accel X: -0.013 Accel Y: 0.012 Accel Z: 0.922 Gyro X: 0.000 Gyro Y: -0.000 Gyro Z:
-0.000
Accel X: -0.012 Accel Y: 0.011 Accel Z: 0.939 Gyro X: 0.000 Gyro Y: -0.000 Gyro Z:
-0.000
Accel X: -0.016 Accel Y: 0.009 Accel Z: 0.931 Gyro X: 0.000 Gyro Y: -0.000 Gyro Z:
-0.000
Accel X: -0.017 Accel Y: 0.012 Accel Z: 0.933 Gyro X: 0.000 Gyro Y: -0.000 Gyro Z:
-0.000
Accel X: -0.017 Accel Y: 0.014 Accel Z: 0.941 Gyro X: 0.000 Gyro Y: -0.000 Gyro Z:
-0.000
```

```
Accel X: -0.015 Accel Y: 0.016 Accel Z: 0.922 Gyro X: 0.000 Gyro Y: -0.000 Gyro Z:
-0.000
Accel X: -0.017 Accel Y: 0.004 Accel Z: 0.934 Gyro X: 0.000 Gyro Y: -0.000 Gyro Z:
-0.000
Accel X: -0.018 Accel Y: 0.008 Accel Z: 0.930 Gyro X: 0.000 Gyro Y: -0.000 Gyro Z:
-0.000
Accel X: -0.022 Accel Y: 0.007 Accel Z: 0.932 Gyro X: 0.000 Gyro Y: -0.000 Gyro Z:
-0.000
Stopped
```

Getting Rotation from Degrees per Second

The gyroscope measures degrees per second (how many degrees it has rotated in a second, or speed of rotation). We can plug in deg/sec for rate and seconds for time, so (degrees per second) * (seconds) equals degrees in rotation.

Below is the code showing this:

```
from time import time, sleep
import smbus

bus = smbus.SMBus(1)
previous, current, elapsed = 0, 0, 0
x, y, z = 0, 0, 0

# MPU6050 registers
PWR_MGMT_1   = 0x6B
SMPLRT_DIV   = 0x19
CONFIG       = 0x1A
GYRO_CONFIG  = 0x1B
INT_ENABLE   = 0x38
ACCEL_X      = 0x3B
ACCEL_Y      = 0x3D
ACCEL_Z      = 0x3F
GYRO_X       = 0x43
GYRO_Y       = 0x45
GYRO_Z       = 0x47

device_address = 0x68 # MPU6050 address

def init():
    # Write to sample rate register
    bus.write_byte_data(device_address, SMPLRT_DIV, 7)
    # Write to power management register
    bus.write_byte_data(device_address, PWR_MGMT_1, 1)
    # Write to Configuration register
    bus.write_byte_data(device_address, CONFIG, 0)
    # Write to Gyro configuration register
    bus.write_byte_data(device_address, GYRO_CONFIG, 24)
```

```
    # Write to interrupt enable register
    bus.write_byte_data(device_address, INT_ENABLE, 1)

def read_data(addr):
    # Get elapsed time for calculating gyro angle
    global current, elapsed
    previous = current
    current = time()
    elapsed = current - previous

    # Accel and Gyro value are 16-bit
    high = bus.read_byte_data(device_address, addr)
    low = bus.read_byte_data(device_address, addr + 1)

    # Concatenate higher and lower value
    value = (high << 8) | low

    # Get signed value from sensor
    if value > 32768: value -= 65536
    return value

def scale_gyro(val):
    return val / 131.0

def scale_accel(val):
    return val / 16384.0

def get_rotation():
    x_ang = scale_gyro(read_data(GYRO_X)) # Read gyro data (degrees per second)
    y_ang = scale_gyro(read_data(GYRO_Y))
    z_ang = scale_gyro(read_data(GYRO_Z))

    global x, y, z
    x += round(x_ang * elapsed * 1000) # Degrees/second multiplied by seconds =
degrees
    y += round(y_ang * elapsed * 1000) # Round values to prevent drift
    z += round(z_ang * elapsed * 1000)

init()
while True:
    try:
        get_rotation()
        print("X: %d Y: %d Z: %d" % (x, y, z))
        sleep(0.2)
    except KeyboardInterrupt:
        print("Stopped")
        break
```

Output

```
X: 19 Y: 0 Z: -5
X: 27 Y: -1 Z: -5
X: 34 Y: -4 Z: -4
X: 40 Y: -6 Z: -3
X: 46 Y: -7 Z: -4
X: 53 Y: -7 Z: -4
X: 60 Y: -6 Z: -5
X: 50 Y: -6 Z: -6
X: 40 Y: -6 Z: -7
X: 31 Y: -2 Z: -9
X: 18 Y: -3 Z: -7
```