Contents

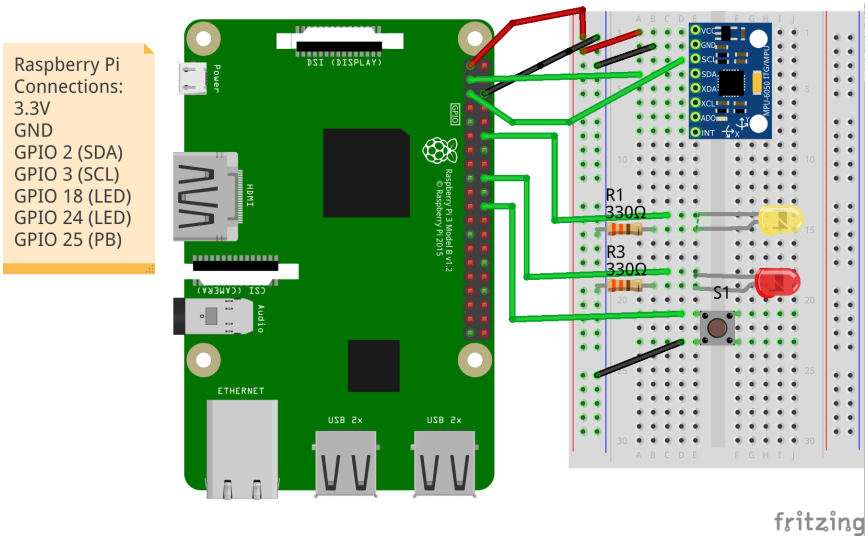
1 Beginning Python with Raspberry Pi Week 3 (Gyro)

1.1 This weeks Circuit: I2C and a Gyroscope (Invensense MPU-6050) + LED and Push button

This week we are going to add a gyroscope to last weeks circuit and show you how read from the gyro using python. The Raspberry Pi talks to the Gyro using a two wire interface called "I2C" which is very common. We will

show you how to enable your I2C interface and test that is working before running your program

2 Build the Circuit



3 Using I2C on Raspberry Pi (May already be done)

We need to setup a few things

3.1 Enable I2C (raspi-config)

You can to this through the gui or command line. Since most of the raspberry pi's I use don't have am monitor I will show the command line version sudo raspi-config arrow down to advanced options and hit enter arrow down to I2C and hit enter select yes arrow over to select finish

3.2 Install i2c-tools (i2cdetect)

sudo apt-get install i2ctools which i2c-tools

4 Install SW for mpu6050 and python (May already be done)

4.1 Install Dependencies for MPU6050 lib

sudo apt install python-smbus sudo pip install mpu6050-raspberrypi

5 Test the circuit

Power up the raspberry pi, open a terminal and type sudo i2cdetect 1 Select Y Expected output—>

0 1 2 3 4 5 6 7 8 9 a b c d e f 00: – – – – – – – – – – – – – – – – 10: – – – – – – – – – – – – – – – – 20: – – – – – – – – – – – – – – – – 30: – – – – – – – – – – – – – – – – 40: – – – – – – – – – – – – – – – – 50: – – – – – – – – – – – – – – – – 60: – – – – – – – – 68 – – – – – – – – 70: – – – – – – – –

6 Test the gyro

Start python interpreter and type the following lines

```
>>> from mpu6050 import mpu6050
>>> sensor = mpu6050(0x68)
>>> accelerometer_data = sensor.get_accel_data()
### First tell the shell you want to use LED
>>> from gpiozero import LED
### Make a red LED
>>> red = LED(18)
### Turn it on then off
>>> red.on()
>>> red.off()
### Repeat it with the yellow LED
>>> yellow = LED(24)
>>> yellow.on()
>>> yellow.off()
### Test the button.
>>> from gpiozero import Button
>>> button = Button(25)
>>> button.is_pressed
False
```

```
### Hold down the button while you press enter
>>> button.is_pressed
True
>>>
```

7 Look at the example

Open the file turtleracewithgyro.py in the "Week3" folder on your desktop This program is more complicated than the previous week. A few things, like user defined functions, global variables, and timers to automatically call functoins periodically

8 user defined functions, i.e. def startRace()

```
def startRace():
    global runRace
    global speed
    runRace = True
    speed = 0
    ada.goto(-160, 00)
    bob.goto(-160, 70)
    ada.clear()
    bob.clear()
    bob.seth(randint(1, 360))  # start out in Random direction
    red.off()
    yellow.off()
```

9 Global Variables, example runRace

runRace is the same varibale in all three cases. Using global runRace makes sure that is the case

```
runRace = False
def startRace():
    global runRace
    global speed
    runRace = True
...
    yellow.off()
```

```
...
def race():
global runRace
if(runRace):
    bob.seth(heading)  # Update Bob's heading
...
# Call update the race every 100 ms
turtle.getscreen().ontimer(race, 100)
```

10 Turtle Graphic Timers to call functions repeatedly at a fixed interval

```
def updateGyro():
...
turtle.getscreen().ontimer(updateGyro, 50)  # call again in 50 ms

def race():
global runRace
if(runRace):
    bob.seth(heading)  # Update Bob's heading
...
# Call update the race every 100 ms
turtle.getscreen().ontimer(race, 100)  # call again in 100 ms

updateGyro()
race()
turtle.mainloop() # Starts all the program running
```

11 run the example

11.1 Press the button to start

11.2 Tilt the gyro left/right to turn the turtle

11.3 Tilt the gyro forward/backward to change the speed

11.4 See if you can get Bob to beat Ada!