

**Department of Computer Science and Engineering**

| **Course Code: CSE461** | **Credits: 1.5** |
| --- | --- |
| **Course Name: Introduction to Robotics Lab** | **Semester: Summer ‘22** |

**Lab 5**

**Controlling of DC Motors using MPU-92/65- a 9-axis motion processing unit**

# I. Topic Overview:

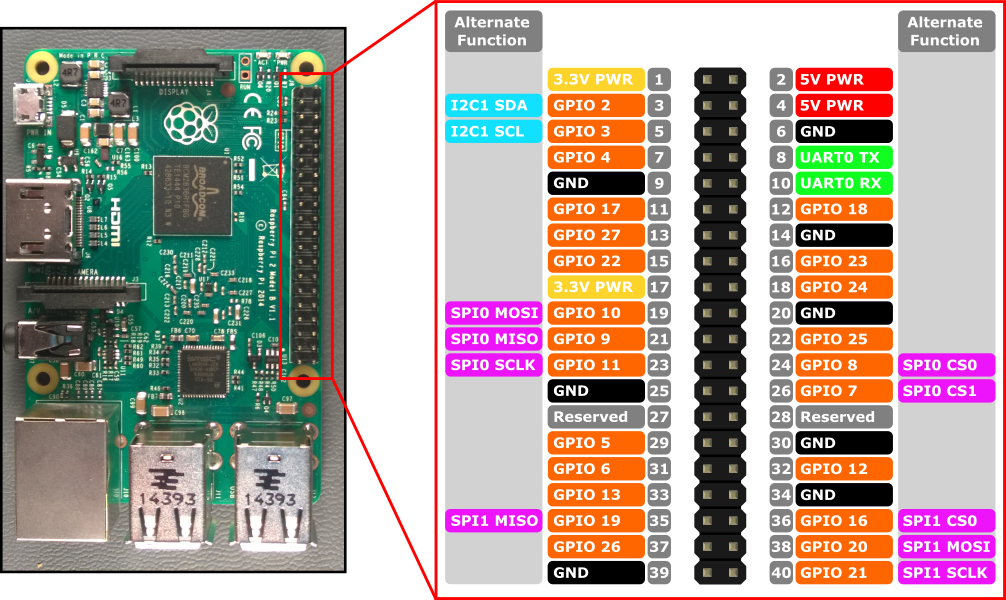
This Lab is designed to provide the basic idea of how to interface an IMU sensor to a Raspberry Pi and read the values from the accelerometer, gyroscope, and magnetometer. Students are going to control a DC motor using the MPU-92/65. By moving the sensor in an upward or downward direction, the direction of the motors will be changed.

# III. Learning Outcome:

After this lecture, the students will be able to learn:

1. Theory and principles of an IMU sensor
2. How to use IMU sensor and how to read the values using I2C
3. How to control a DC motor using l298N Motor driver and the value received from IMU

The Raspberry Pi 4 has 40 GPIO pins that can be easily configured to read inputs or write outputs.



Here, you will be able to learn the functioning of each pin, which helps you to do things on your Raspberry Pi 4 easily. There are 40 pins in this model and among them 26 are GPIO pins.

**Required Hardware:**

* Raspberry Pi
* MPU 92/65 IMU
* DC Motor (1 piece)
* Motor Driver L298N
* Battery 3.7V (2 piece)
* Battery Case (1)
* Jumper Wire
* Breadboard

### What is IMU:

Inertial Measurement Units (IMU) are sensors used to understand the orientation of a body in three-dimensional spacee.Once attached to an object, IMU sensors can provide information on the body’s angular rate, orientation, and even force applied to it by measuring the linear and angular motion using the combination of accelerometer, gyroscope, and magnetometer.

***Gyroscope***

A gyroscope (sometimes simply called a gyro) measures the rate of rotation and tracks it over time to calculate the current angle. As it tracks the rate of rotation, the gyro drifts. Gyros work well for measuring quick, sharp movements.

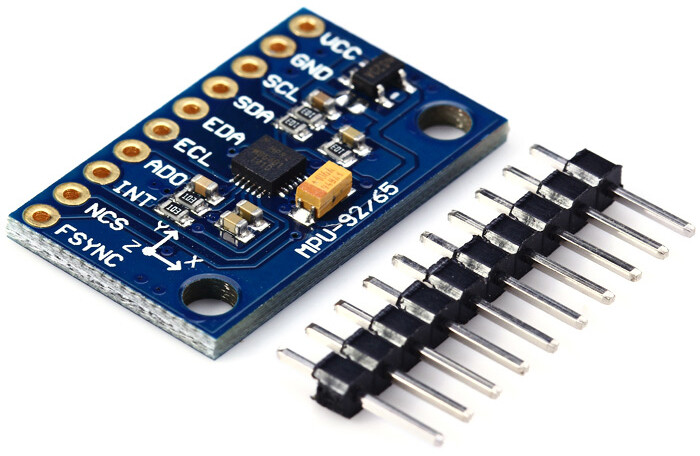
***Accelerometers***

Accelerometers are real-time compared to gyros as they don’t have to be tracked all the time, they can give the angle values at any moment. They are useful in measuring both gravity based static movements as well as more dynamic movements based on inertia and acceleration. The downside is that they can get noisy, so they need to be used for measuring angles over a period of time.

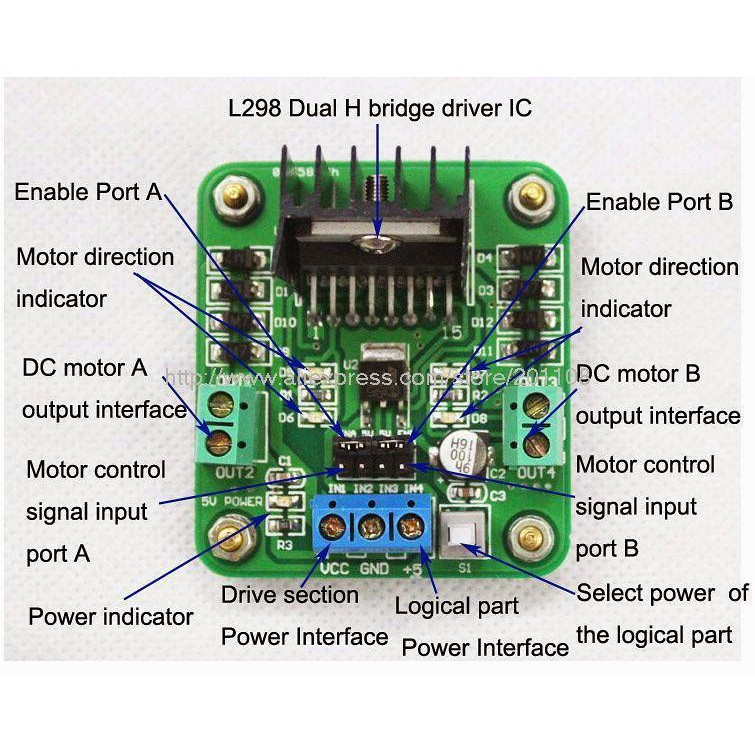
***Magnetometer***

A magnetometer uses the earth’s magnetic field to understand the direction. In IMUs (Inertial Measurement Units) they are mostly used in combination with an accelerometer and a gyroscope in order to compensate for the long term drift in sensor data from the gyro and to correct the gyro bias.

We are using the MPU 92/65 Model

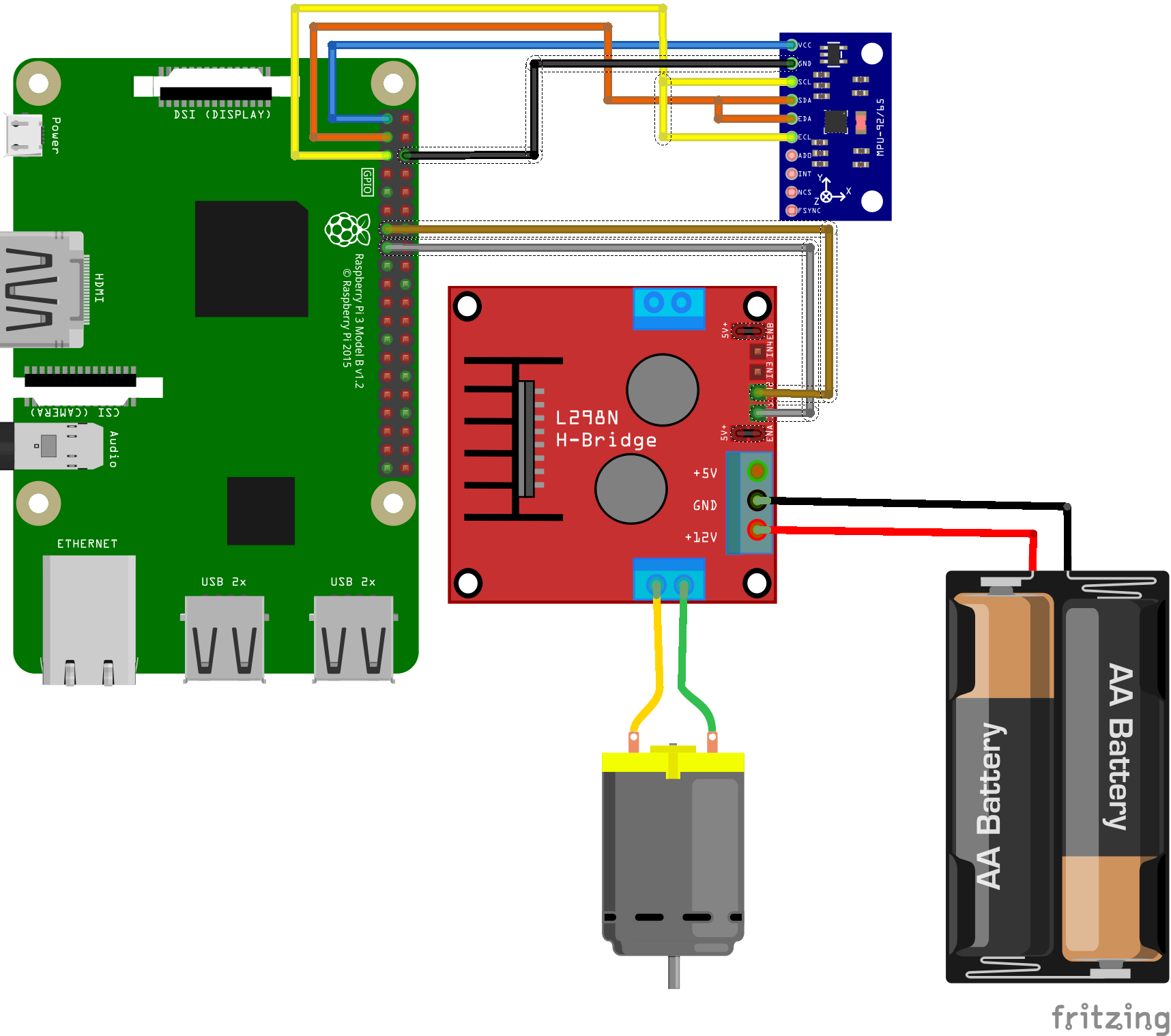


**Motor Driver L298N**



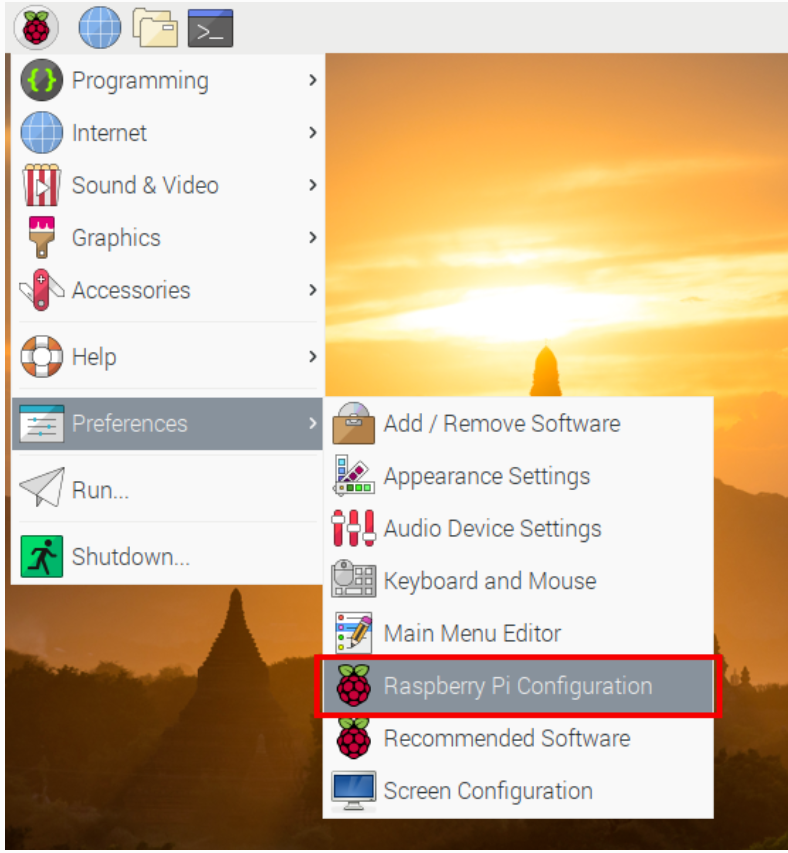
L298N module is a high voltage, high current dual full-bridge motor driver module for controlling DC motor and stepper motor. It can control both the speed and rotation direction of two DC motors. This module consists of an L298 dual-channel H-Bridge motor driver IC. This module uses two techniques for the control speed and rotation direction of the DC motors. These are PWM – For controlling the speed and H-Bridge – For controlling rotation direction. These modules can control two DC motor or one stepper motor at the same time.

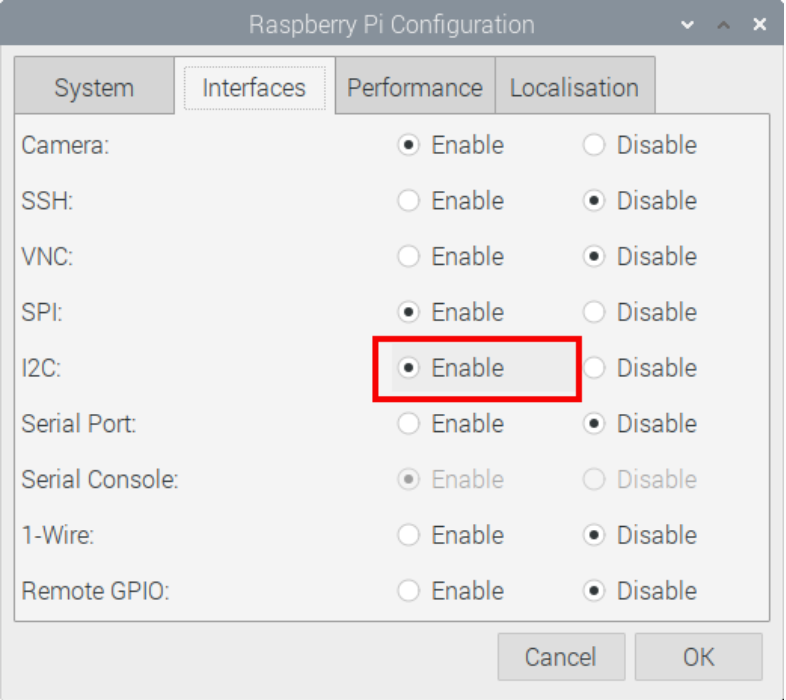
### Circuit diagram of the setup:

****

**SETUP :**

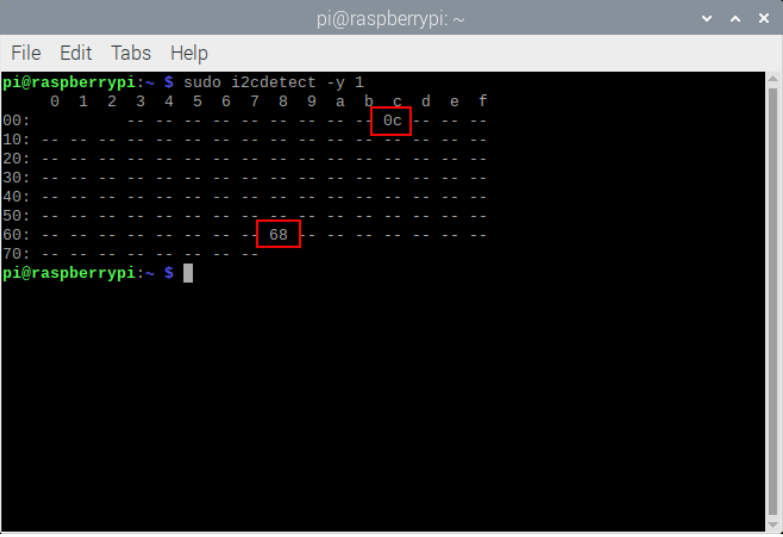
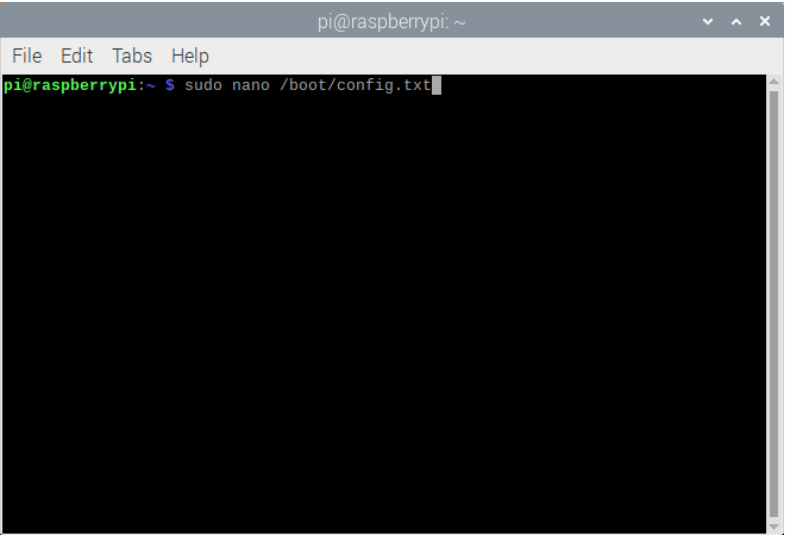
A Raspberry Pi will be used to read the MPU92/65 3-axis acceleration, 3-axis angular rotation speed, and 3-axis magnetic flux. The MPU92/65 will communicate with the Raspberry Pi using the I2C protocol. In order to read and write data via I2C, we must first enable the I2C ports on the RPi.

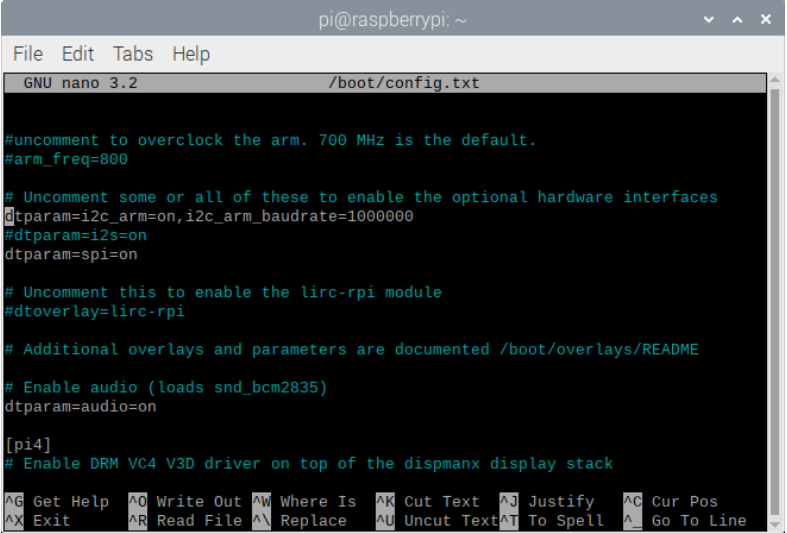
1. Raspberry Pi Configuration
2. Enable I2C



1. Open Command Window and type “sudo i2cdetect -y 1”

Verify 0x68 (MPU6050) and 0x0C (AK8963) as I2C devices

1. sudo nano /boot/config.txt
2. Add Line in Next to I2C Setting



**Interfacing the MPU9250 with Python**

\*\*The code block given below handles the startup for each I2C sensor (MPU6050 and AK8963) and also the conversion from bits to real-world values (gravitation, degrees per second, and Teslas). The code block should be saved in the local folder under the name ‘mpu9250\_i2c.py’ - this library will be imported in the example below.

| ***# this is to be saved in the local folder under the name "mpu9250\_i2c.py"***  ***# it will be used as the I2C controller and function harbor for the project***  ***# refer to datasheet and register map for full explanation***  **import smbus,time**  **def MPU6050\_start():**  ***# alter sample rate (stability)***  **samp\_rate\_div = 0 *# sample rate = 8 kHz/(1+samp\_rate\_div)***  **bus.write\_byte\_data(MPU6050\_ADDR, SMPLRT\_DIV, samp\_rate\_div)**  **time.sleep(0.1)**  ***# reset all sensors***  **bus.write\_byte\_data(MPU6050\_ADDR,PWR\_MGMT\_1,0x00)**  **time.sleep(0.1)**  ***# power management and crystal settings***  **bus.write\_byte\_data(MPU6050\_ADDR, PWR\_MGMT\_1, 0x01)**  **time.sleep(0.1)**  ***#Write to Configuration register***  **bus.write\_byte\_data(MPU6050\_ADDR, CONFIG, 0)**  **time.sleep(0.1)**  ***#Write to Gyro configuration register***  **gyro\_config\_sel = [0b00000,0b010000,0b10000,0b11000] *# byte registers***  **gyro\_config\_vals = [250.0,500.0,1000.0,2000.0] *# degrees/sec***  **gyro\_indx = 0**  **bus.write\_byte\_data(MPU6050\_ADDR, GYRO\_CONFIG, int(gyro\_config\_sel[gyro\_indx]))**  **time.sleep(0.1)**  ***#Write to Accel configuration register***  **accel\_config\_sel = [0b00000,0b01000,0b10000,0b11000] *# byte registers***  **accel\_config\_vals = [2.0,4.0,8.0,16.0] *# g (g = 9.81 m/s^2)***  **accel\_indx = 0**  **bus.write\_byte\_data(MPU6050\_ADDR, ACCEL\_CONFIG, int(accel\_config\_sel[accel\_indx]))**  **time.sleep(0.1)**  ***# interrupt register (related to overflow of data [FIFO])***  **bus.write\_byte\_data(MPU6050\_ADDR, INT\_ENABLE, 1)**  **time.sleep(0.1)**  **return gyro\_config\_vals[gyro\_indx],accel\_config\_vals[accel\_indx]**    **def read\_raw\_bits(register):**  ***# read accel and gyro values***  **high = bus.read\_byte\_data(MPU6050\_ADDR, register)**  **low = bus.read\_byte\_data(MPU6050\_ADDR, register+1)**  ***# combine higha and low for unsigned bit value***  **value = ((high << 8) | low)**    ***# convert to +- value***  **if(value > 32768):**  **value -= 65536**  **return value**  **def mpu6050\_conv():**  ***# raw acceleration bits***  **acc\_x = read\_raw\_bits(ACCEL\_XOUT\_H)**  **acc\_y = read\_raw\_bits(ACCEL\_YOUT\_H)**  **acc\_z = read\_raw\_bits(ACCEL\_ZOUT\_H)**  ***# raw temp bits***  ***## t\_val = read\_raw\_bits(TEMP\_OUT\_H) # uncomment to read temp***    ***# raw gyroscope bits***  **gyro\_x = read\_raw\_bits(GYRO\_XOUT\_H)**  **gyro\_y = read\_raw\_bits(GYRO\_YOUT\_H)**  **gyro\_z = read\_raw\_bits(GYRO\_ZOUT\_H)**  ***#convert to acceleration in g and gyro dps***  **a\_x = (acc\_x/(2.0\*\*15.0))\*accel\_sens**  **a\_y = (acc\_y/(2.0\*\*15.0))\*accel\_sens**  **a\_z = (acc\_z/(2.0\*\*15.0))\*accel\_sens**  **w\_x = (gyro\_x/(2.0\*\*15.0))\*gyro\_sens**  **w\_y = (gyro\_y/(2.0\*\*15.0))\*gyro\_sens**  **w\_z = (gyro\_z/(2.0\*\*15.0))\*gyro\_sens**  ***## temp = ((t\_val)/333.87)+21.0 # uncomment and add below in return***  **return a\_x,a\_y,a\_z,w\_x,w\_y,w\_z**  **def AK8963\_start():**  **bus.write\_byte\_data(AK8963\_ADDR,AK8963\_CNTL,0x00)**  **time.sleep(0.1)**  **AK8963\_bit\_res = 0b0001 *# 0b0001 = 16-bit***  **AK8963\_samp\_rate = 0b0110 *# 0b0010 = 8 Hz, 0b0110 = 100 Hz***  **AK8963\_mode = (AK8963\_bit\_res <<4)+AK8963\_samp\_rate *# bit conversion***  **bus.write\_byte\_data(AK8963\_ADDR,AK8963\_CNTL,AK8963\_mode)**  **time.sleep(0.1)**    **def AK8963\_reader(register):**  ***# read magnetometer values***  **low = bus.read\_byte\_data(AK8963\_ADDR, register-1)**  **high = bus.read\_byte\_data(AK8963\_ADDR, register)**  ***# combine higha and low for unsigned bit value***  **value = ((high << 8) | low)**  ***# convert to +- value***  **if(value > 32768):**  **value -= 65536**  **return value**  **def AK8963\_conv():**  ***# raw magnetometer bits***  **loop\_count = 0**  **while 1:**  **mag\_x = AK8963\_reader(HXH)**  **mag\_y = AK8963\_reader(HYH)**  **mag\_z = AK8963\_reader(HZH)**  ***# the next line is needed for AK8963***  **if bin(bus.read\_byte\_data(AK8963\_ADDR,AK8963\_ST2))=='0b10000':**  **break**  **loop\_count+=1**    ***#convert to acceleration in g and gyro dps***  **m\_x = (mag\_x/(2.0\*\*15.0))\*mag\_sens**  **m\_y = (mag\_y/(2.0\*\*15.0))\*mag\_sens**  **m\_z = (mag\_z/(2.0\*\*15.0))\*mag\_sens**  **return m\_x,m\_y,m\_z**    ***# MPU6050 Registers***  **MPU6050\_ADDR = 0x68**  **PWR\_MGMT\_1 = 0x6B**  **SMPLRT\_DIV = 0x19**  **CONFIG = 0x1A**  **GYRO\_CONFIG = 0x1B**  **ACCEL\_CONFIG = 0x1C**  **INT\_ENABLE = 0x38**  **ACCEL\_XOUT\_H = 0x3B**  **ACCEL\_YOUT\_H = 0x3D**  **ACCEL\_ZOUT\_H = 0x3F**  **TEMP\_OUT\_H = 0x41**  **GYRO\_XOUT\_H = 0x43**  **GYRO\_YOUT\_H = 0x45**  **GYRO\_ZOUT\_H = 0x47**  ***#AK8963 registers***  **AK8963\_ADDR = 0x0C**  **AK8963\_ST1 = 0x02**  **HXH = 0x04**  **HYH = 0x06**  **HZH = 0x08**  **AK8963\_ST2 = 0x09**  **AK8963\_CNTL = 0x0A**  **mag\_sens = 4900.0 *# magnetometer sensitivity: 4800 uT***  ***# start I2C driver***  **bus = smbus.SMBus(1) *# start comm with i2c bus***  **gyro\_sens,accel\_sens = MPU6050\_start() *# instantiate gyro/accel***  **AK8963\_start() *# instantiate magnetometer*** |
| --- |

The code is given below, along with the sample readouts printed to the Python console:

| from mpu9250\_i2c import \*  time.sleep(1) # delay necessary to allow mpu9250 to settle  print('recording data')  while 1:  try:  ax,ay,az,wx,wy,wz = mpu6050\_conv() # read and convert mpu6050 data  mx,my,mz = AK8963\_conv() # read and convert AK8963 magnetometer data  except:  continue    print('{}'.format('-'\*30))  print('accel [g]: x = {0:2.2f}, y = {1:2.2f}, z {2:2.2f}= '.format(ax,ay,az))  print('gyro [dps]: x = {0:2.2f}, y = {1:2.2f}, z = {2:2.2f}'.format(wx,wy,wz))  print('mag [uT]: x = {0:2.2f}, y = {1:2.2f}, z = {2:2.2f}'.format(mx,my,mz))  print('{}'.format('-'\*30))  time.sleep(1) |
| --- |

The following code run the motors based on gyro value:

| from mpu9250\_i2c import \*  time.sleep(1) # delay necessary to allow mpu9250 to settle  import RPi.GPIO as GPIO  from time import sleep  in1 = 27  in2 = 22  GPIO.setmode(GPIO.BCM)  GPIO.setup(in1,GPIO.OUT)  GPIO.setup(in2,GPIO.OUT)  GPIO.output(in1,GPIO.LOW)  GPIO.output(in2,GPIO.LOW)  print("\n")  print("The default speed & direction of motor is LOW & Forward.....")  print("r-run s-stop f-forward b-backward l-low m-medium h-high e-exit")  print("\n")  print('recording data')  while 1:  try:  ax,ay,az,wx,wy,wz = mpu6050\_conv() # read and convert mpu6050 data  mx,my,mz = AK8963\_conv() # read and convert AK8963 magnetometer data  except:  continue    if -10<wx<10:  print("stop")  GPIO.output(in1,GPIO.LOW)  GPIO.output(in2,GPIO.LOW)  print("%.2f",wx)  elif wx >10:  print("run")  GPIO.output(in1,GPIO.HIGH)  GPIO.output(in2,GPIO.LOW)  print("%.2f",wx)    elif wx<-10:  print("backward")  GPIO.output(in1,GPIO.LOW)  GPIO.output(in2,GPIO.HIGH)  print("%.2f",wx)  else:  print("<<< wrong data >>>")  print("please enter the defined data to continue.....") |
| --- |

**Lab Task:**

Now your task is to modify the code in such a way that you can implement following task:

* Take input from keyboard using **input()** function and turn on & off the Motor.
* Rotate motor Clockwise if the y axis value is greater than +15 and rotate the motor anti clockwise if the y axis gyro value is less than -10.