All that begins …

السلام عليكم

peace be upon you

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

OSCAE.Initiative
(Open Source Computer Aided Engineering)

Raspberry Pi 3 for Scientific Computing

An Introduction

Abu Hasan 'ABDULLAH

February 2019

Introduction

- For scientific computing environment on RPi3 B+, **Raspbian** OS is recommended for being relatively fast. It comes with a healthy collection of tools, especially tools for developing software, allowing user to straight away jump into it after installation. You will need:
    1. Micro SD card preinstalled with **Raspbian** OS and case
    2. Micro USB power cable: RPi3 requires a 2.5A power supply
    3. HDMI to VGA converter or HDMI cable

(a) RPi3 B+

(b) Micro SD

(c) Case

(d) Power adapter

(e) Video converter

4. Monitor or TV: If your monitor is DVI or VGA, you will need an adapter.
5. USB mouse & keyboard

Raspbian OS comes standard with the following packages:

- **Programming**
  1. BlueJ Java IDE
  2. Geany Programmer's Editor
  3. Greenfoot Jave IDE
  4. Mathematica
  5. Node-RED
  6. Python 3 (IDLE)
  7. Scratch
  8. Sense HAT Emulator
  9. Sonic Pi
  10. Thonny Python IDE
  11. Wolfram
- **Office Tools**
  1. LibreOffice
- **Internet Tools**
  1. Chromium
  2. Claws Mail
  3. VNC Viewer
- **Accessories**
  1. Archiver
  2. Calculator
  3. File Manager
  4. Image Viewer
  5. PDF Viewer
  6. SD Card Copier
  7. Task Manager
  8. Terminal
  9. Text Editor

On top of those standard desktop computing tools, we added three more categories of software popular among users within the scientific & engineering communities.

- **CAE & Scientific**
  1. FreeCAD, Blender, LibreCAD & KiCAD
  2. gmsh, Netgen & MeshLab
  3. GNU Octave, Maxima
  4. g3data, GNUplot & SciDAVis
  5. R & GNU PSPP
- **Programming & SDK**
  1. Arduino
  2. GNU SDK (C, C++, Fortran), OpenMPI, f2c, fort77, ftnchek
  3. Python for Math/Science/Engineering: SciPy, NumPy, Matplotlib, Pandas
  4. Python programming utils: IDLE 3, ipython, Spyder
  5. Tcl/Tk/Tix, Qt4 Designer, FLTK, VTK, wxGTK, CERNLIB, PETSc
  6. CodeBlock, CodeLite, CMake, gedit, Meld, rdiff, xxdiff
- **TEX/LATEX-based typesetting tools**
  1. Full TEXLive installation
  2. TEXworks
  3. JabRef

Writing & Running Octave/Matlab Codes

- Matlab does NOT have a native version running on Raspberry Pi but GNU Octave is the best open-source alternative to Matlab.
- To launch Octave with its GUI (where you can enter Octave/Matlab commands just like the command line) enter:

```
$ octave --force-gui
```

- If you want to perform Octave-based analytics in a batch processing environment, you need to be able to run Octave scripts from the command line—the process consists of three steps:
  1. Writing an Octave program
  2. Running an Octave program
  3. Making an Octave program executable

- To demonstrate how to create an Octave program, and run it on the Raspberry Pi, we'll make a simple program that will print "Salaam, World!" in the terminal.

1 Writing an Octave program

- To start, open the nano text editor and create a new file with a ".m" extension by entering this at the command prompt:

```
nano salaam.m
```

This file is where you'll write the Octave code. You can write the code in any text editor, just make sure to give the file a .m extension.

- Now, enter this code into nano:

```
#!/usr/bin/octave -qf

S = 'Salaam, World!';
disp(S)
```

After entering the code, enter Ctrl-X and Y to save and exit nano.

2 Running an Octave program

- To run the program without making it executable, navigate to the location where you saved your file, and enter this at the command prompt:

```
octave-cli salaam.m
```

3 Making a Octave program executable

- Making a Octave program executable allows you to run the program without entering `octave-cli` before the file name. You can make a file executable by entering this at the command prompt:

```
chmod +x  salaam.m
```

Now to run the program, all you need to enter is:

```
./salaam.m
```

Writing & Running GNU C Codes

- To demonstrate how to create a C program, compile it, and run it on the Raspberry Pi, we'll make a simple program that will print “Salaam, World!” in the terminal.
- The coding process in C consists of four steps:
  1. Creating the C source file
  2. Compiling the C source file into a program
  3. Making the program executable
  4. Executing the program

1 Creating the C source file

- To start, open the nano text editor and create a new file with a ".c" extension by entering this at the command prompt:

```
nano salaam.c
```

This file is where you'll write the C code. You can write the code in any text editor, just make sure to give the file a .c extension.

- Now, enter this code into nano text editor:

```
#include <stdio.h>

int main()
{
   printf("Salaam, World! \n");
   return 0;
}
```

After entering the code, enter Ctrl-X and Y to save and exit nano.

2. **Compiling the C source file into a program**
   - Code written in C will need to be compiled before it can be run on a computer. Compiling is the process of converting the code you write into machine readable instructions that can be understood by the computer's processor.
   - When you compile your source file, a new compiled file gets created. For example, entering the command below will compile `salaam.c` into a new file called `mysalaam`:

```
gcc salaam.c -o mysalaam
```

3. **Making the program executable**
   - Now we need to make the compiled file executable. To do that, we just need to change the file permissions. Enter this at the command prompt:

```
chmod +x  mysalaam
```

4. **Executing the program**
   - Now all we need to do to run the compiled, executable, `C` program is enter this at the command prompt:

```
./mysalaam
```

Writing & Running GNU Fortran Codes

- We will follow steps we took earlier to develop a program using the C programming language. We'll make a simple program that will print "Salaam, World!" in the terminal.
- The coding process in Fortran also consists of four steps:
    1. Creating the Fortran source file
    2. Compiling the Fortran source file into a program
    3. Making the program executable
    4. Executing the program

1 Creating the Fortran source file

- To start, open the nano text editor and create a new file with a ".f90" extension by entering this at the command prompt:

```
nano salaam.f90
```

This file is where you'll write the Fortran code. You can write the code in any text editor, just make sure to give the file a .f90 extension.

- Now, enter this code into nano text editor:

```
program salaam
   print *, "Salaam, World!"
end program salaam
```

Fortran is case insensitive, one could just as easily write the first salaam.f90 program as:

```
Program Salaam
   Print *, "Salaam, World!"
End Program Salaam
```

After entering the code, enter Ctrl-X and Y to save and exit nano.

2 Compiling the Fortran source file into a program

- As in C, code written in Fortran will need to be compiled before it can be run on a computer. Compiling is the process of converting the code you write into machine readable instructions that can be understood by the computer's processor.

- When you compile your source file, a new compiled file gets created. For example, entering the command below will compile `salaam.f90` into a new file called `mysalaam`:

```
gfortran salaam.f90 -o mysalaam
```

3 Making the program executable

- Now we need to make the compiled file executable. To do that, we just need to change the file permissions. Enter this at the command prompt:

```
chmod +x  mysalaam
```

4 Executing the program

- Now all we need to do to run the compiled, executable, `Fortran` program is enter this at the command prompt:

```
./mysalaam
```

Writing & Running Python Codes

- Unlike C programs, Python programs don't need to be compiled before running them. However, you will need to install the *Python interpreter* on your computer to run them. The Python interpreter is a program that reads Python files and executes the code.
- A *Read-Eval-Print Loop* (REPL) is a simple, interactive computer programming environment that takes user's inputs, evaluates them, and returns the result to the user.
- To access the Python REPL (where you can enter Python commands just like the command line) enter `python` or `python3` depending on which version you want to use:

```
python
```

or

```
python3
```

- To demonstrate how to create a Python program, and run it on the Raspberry Pi, we'll make a simple program that will print "Salaam, World!" in the terminal.
- The coding process in Python consists of three steps:
  1. Writing a Python program
  2. Running a Python program
  3. Making a Python program executable

1. **Writing a Python program**

   - To start, open the `nano` text editor and create a new file with a "`.py`" extension by entering this at the command prompt:

   ```
   nano salaam.py
   ```

   This file is where you'll write the Python code. You can write the code in any text editor, just make sure to give the file a `.py` extension.

   - Now, enter this code into `nano`:

   ```
   #!/usr/bin/python

   print "Salaam, World!"
   ```

   After entering the code, enter `Ctrl-X` and `Y` to save and exit `nano`.

2. **Running a Python program**

   - To run the program without making it executable, navigate to the location where you saved your file, and enter this at the command prompt:

   ```
   python salaam.py
   ```

3 Making a Python program executable

- Making a Python program executable allows you to run the program without entering `python` before the file name. You can make a file executable by entering this at the command prompt:

```
chmod +x  salaam.py
```

Now to run the program, all you need to enter is:

```
./salaam.py
```

Raspberry Pi based Data Loggers

Raspberry Pi based Data Loggers

GPIO Pins

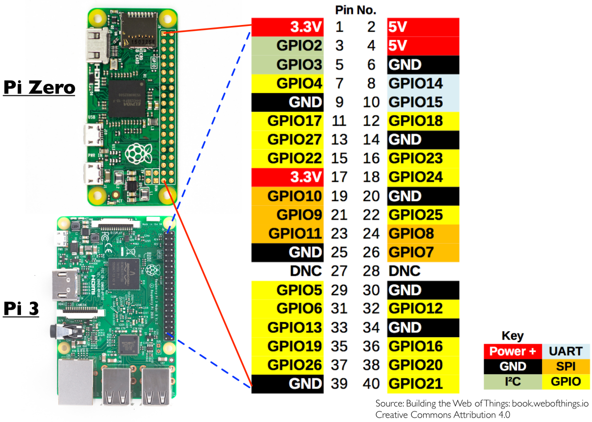


Figure 1: GPIO pinout for Raspberry Pi 3 and Zero.

https://learn.sparkfun.com/tutorials/raspberry-gpio/gpio-pinout

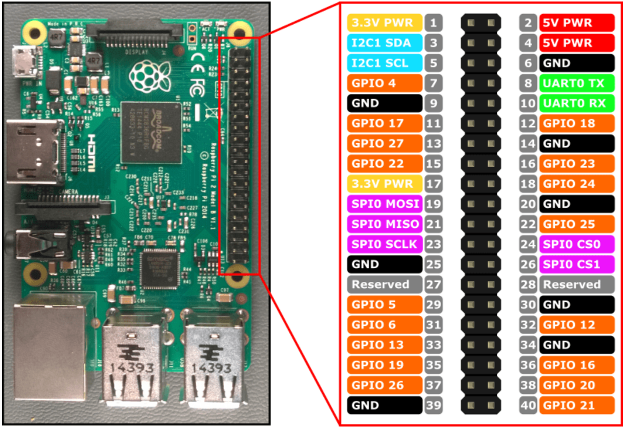


Figure 2: GPIO pinout for Raspberry Pi 2.

https://learn.sparkfun.com/tutorials/raspberry-gpio/gpio-pinout

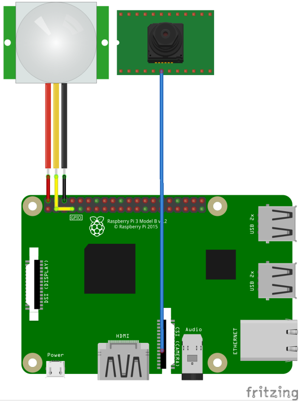


Figure 3: PIR motion sensor with surveillance camera.

https://edi.wang/post/2016/8/11/raspi-azure-camera

Sample code

```
#!/usr/bin/python
#
import RPi.GPIO as GPIO
import time

from picamera import PiCamera

pirPin = 7

GPIO.setmode(GPIO.BOARD)
GPIO.setup(pirPin, GPIO.IN)

camera = PiCamera()
counter = 1

while True:
    if GPIO.input(pirPin):
    print "Motion detected!"

        try:
        timestr = time.strftime("%Y%m%d-%H%M%S")
        print timestr
        camera.start_preview()
        time.sleep(1)
        camera.capture('/home/pi/media/0-pix/Picture-%s.jpg' % timestr)
        counter = counter + 1
        camera.stop_preview()

    except:
        camera.stop_preview()

    time.sleep(2)
```

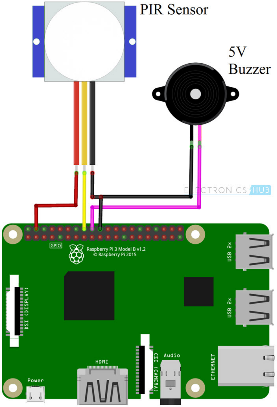
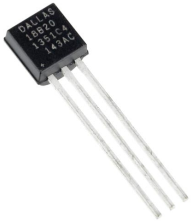


Figure 4: PIR motion sensor with alarm.

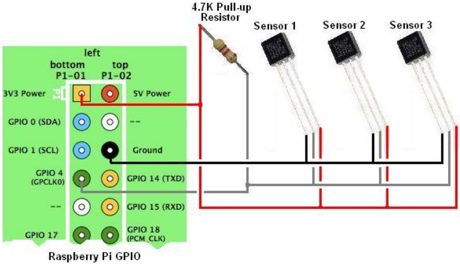
https://www.electronicshub.org/pir-motion-sensor-using-raspberry-pi/

Sample code

```
import RPi.GPIO as GPIO
import time

sensor = 16
buzzer = 18

GPIO.setmode(GPIO.BOARD)
GPIO.setup(sensor,GPIO.IN)
GPIO.setup(buzzer,GPIO.OUT)

GPIO.output(buzzer,False)
print "Initialzing PIR Sensor......"
time.sleep(12)
print "PIR Ready..."
print " "

try:
   while True:
      if GPIO.input(sensor):
          GPIO.output(buzzer,True)
          print "Motion Detected"
          while GPIO.input(sensor):
              time.sleep(0.2)
      else:
          GPIO.output(buzzer,False)


except KeyboardInterrupt:
GPIO.cleanup()
```



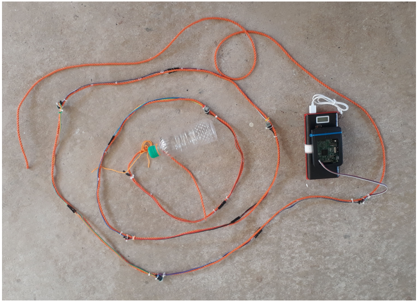
(a) Sensor



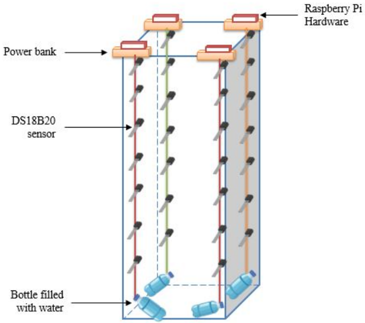
(b) Wiring

Figure 5: Connecting multiple DS18B20 temperature sensors in series.

http://www.reuk.co.uk/wordpress/raspberry-pi/connect-multiple-temperature-sensors-with-raspberry-pi/
http://www.reuk.co.uk/wordpress/raspberry-pi/raspberry-pi-temperature-logger-with-xively/



(a) DAQ for earth cooler



(b) Instrumenting the earth cooler

Figure 6: Temperature monitoring inside an earth cooler.

Farah Atiqah binti Ibrahim, A. H. Abdullah, N. Kamaruzaman (2018): *Development of a Simple Data Acquisition System for Monitoring Performance of an Earth Cooler*, Faculty of Mechanical Engineering, UTM.

Sample code

```
#!/usr/bin/python
#
import os
import glob
import time
from datetime import datetime
import csv
#import onewire

########################
# Set up all variables #
########################

#allows use of the sensors
os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')
base_dir = '/sys/bus/w1/devices/'

#name the CSV file for the data log
out_filename = '/home/pi/ugp/p08/temperatures.csv'

#initialize variables for counting hot readings
i1 = 0
i2 = 0
i3 = 0
i4 = 0
i5 = 0
i6 = 0
i7 = 0
```

```
#insert device serial numbers here
sn1 = '28-0217b064fdff'
sn2 = '28-0217b066ddff'
sn3 = '28-0217b07248ff'
sn4 = '28-0217b21564ff'
sn5 = '28-0117b10415ff'
sn6 = '28-0217b2877fff'
sn7 = '28-0117b0418cff'

#initialize all of the directories for the sensors
device_file1 = glob.glob(base_dir + sn1)[0] + '/w1_slave'
device_file2 = glob.glob(base_dir + sn2)[0] + '/w1_slave'
device_file3 = glob.glob(base_dir + sn3)[0] + '/w1_slave'
device_file4 = glob.glob(base_dir + sn4)[0] + '/w1_slave'
device_file5 = glob.glob(base_dir + sn5)[0] + '/w1_slave'
device_file6 = glob.glob(base_dir + sn6)[0] + '/w1_slave'
device_file7 = glob.glob(base_dir + sn7)[0] + '/w1_slave'
```

Sample code (continued)

```
################################
# Routines to read each sensor #
################################
#Read Sensor 1
def read_temp_raw1():
    f = open(device_file1, 'r')
    lines1 = f.readlines()
    f.close()
    return lines1

def read_temp1():
    lines1 = read_temp_raw1()
    while lines1[0].strip()[-3:] != 'YES':
time.sleep(0.2)
lines1 = read_temp_raw1()
    equals_pos = lines1[1].find('t=')
    if equals_pos != -1:
temp_string1 = lines1[1][equals_pos+2:]
temp_c1 = float(temp_string1) / 1000.0
return temp_c1
```

```
#Read Sensor 2
def read_temp_raw2():
    f = open(device_file2, 'r')
    lines2 = f.readlines()
    f.close()
    return lines2

def read_temp2():
    lines2 = read_temp_raw2()
    while lines2[0].strip()[-3:] != 'YES':
time.sleep(0.2)
lines2 = read_temp_raw2()
    equals_pos = lines2[1].find('t=')
    if equals_pos != -1:
temp_string2 = lines2[1][equals_pos+2:]
temp_c2 = float(temp_string2) / 1000.0
return temp_c2
```

Sample code (continued)

```
#Read Sensor 3
def read_temp_raw3():
        f = open(device_file3, 'r')
        lines3 = f.readlines()
        f.close()
        return lines3

def read_temp3():
        lines3 = read_temp_raw3()
        while lines3[0].strip()[-3:] != 'YES':
                time.sleep(0.2)
                lines3 = read_temp_raw3()
        equals_pos = lines3[1].find('t=')
        if equals_pos != -1:
                temp_string3 = lines3[1][equals_pos+2:]
                temp_c3 = float(temp_string3) / 1000.0
                return temp_c3
```

```
#Read Sensor 4
def read_temp_raw4():
        f = open(device_file4, 'r')
        lines4 = f.readlines()
        f.close()
        return lines4

def read_temp4():
        lines4 = read_temp_raw4()
        while lines4[0].strip()[-3:] != 'YES':
                time.sleep(0.2)
                lines4 = read_temp_raw4()
        equals_pos = lines4[1].find('t=')
        if equals_pos != -1:
                temp_string4 = lines4[1][equals_pos+2:]
                temp_c4 = float(temp_string4) / 1000.0
                return temp_c4
```

Sample code (continued)

```
#Read Sensor 5
def read_temp_raw5():
        f = open(device_file5, 'r')
        lines5 = f.readlines()
        f.close()
        return lines5

def read_temp5():
        lines5 = read_temp_raw5()
        while lines5[0].strip()[-3:] != 'YES':
                time.sleep(0.2)
                lines5 = read_temp_raw5()
        equals_pos = lines5[1].find('t=')
        if equals_pos != -1:
                temp_string5 = lines5[1][equals_pos+2:]
                temp_c5 = float(temp_string5) / 1000.0
                return temp_c5
```

```
#Read Sensor 6
def read_temp_raw6():
        f = open(device_file6, 'r')
        lines6 = f.readlines()
        f.close()
        return lines6

def read_temp6():
        lines6 = read_temp_raw6()
        while lines6[0].strip()[-3:] != 'YES':
                time.sleep(0.2)
                lines6 = read_temp_raw6()
        equals_pos = lines6[1].find('t=')
        if equals_pos != -1:
                temp_string6 = lines6[1][equals_pos+2:]
                temp_c6 = float(temp_string6) / 1000.0
                return temp_c6
```

Sample code (continued)

```
#Read Sensor 7
def read_temp_raw7():
        f = open(device_file7, 'r')
        lines7 = f.readlines()
        f.close()
        return lines7

def read_temp7():
        lines7 = read_temp_raw7()
        while lines7[0].strip()[-3:] != 'YES':
                time.sleep(0.2)
                lines7 = read_temp_raw7()
        equals_pos = lines7[1].find('t=')
        if equals_pos != -1:
                temp_string7 = lines7[1][equals_pos+2:]
                temp_c7 = float(temp_string7) / 1000.0
                return temp_c7
```

```
#############
# Main Loop #
#############

while True:

    currenttime = time.ctime()
    temp1 = read_temp1()
    temp2 = read_temp2()
    temp3 = read_temp3()
    temp4 = read_temp4()
    temp5 = read_temp5()
    temp6 = read_temp6()
    temp7 = read_temp7()

    with open(out_filename, 'a') as f:
writer = csv.writer(f)
writer.writerow((
    currenttime,

    temp1,
    temp2,
    temp3,
                    temp4,
                    temp5,
                    temp6,
                    temp7,
))
    open(out_filename).close()

    time.sleep(1) #taking reading once per two minute
```

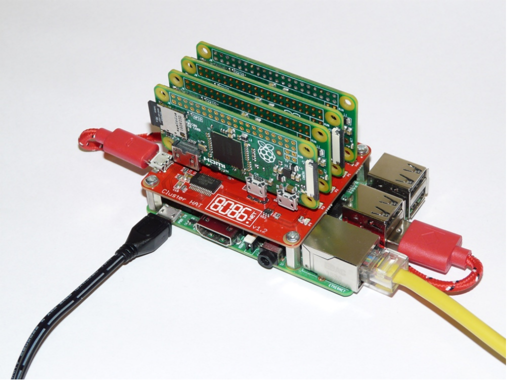
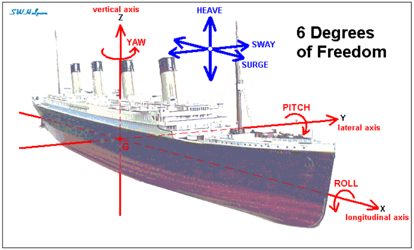
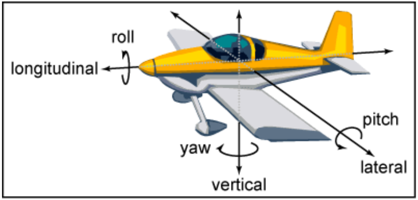


Figure 7: Future expansion for the temperature monitoring system of the earth cooler.

FARAH ATIQAH BINTI IBRAHIM, A. H. ABDULLAH, N. KAMARUZAMAN (2018): *Development of a Simple Data Acquisition System for Monitoring Performance of an Earth Cooler*, Faculty of Mechanical Engineering, UTM.



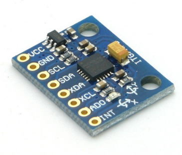
(a) Ship motion



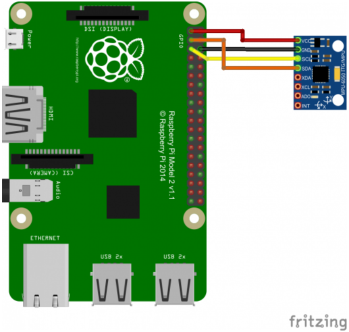
(b) Aircraft motion

Figure 8: Real world 6DOF motions.

http://johnclarkeonline.com/2011/12/18/six-degrees-of-freedom/



(a) Sensor



(b) Wiring

Figure 9: Data acquisition with RPi: 6DOF Motion.

https://tutorials-raspberrypi.com/measuring-rotation-and-acceleration-raspberry-pi/
http://www.electronicwings.com/raspberry-pi/mpu6050-accelerometergyroscope-interfacing-with-raspberry-pi

Sample code

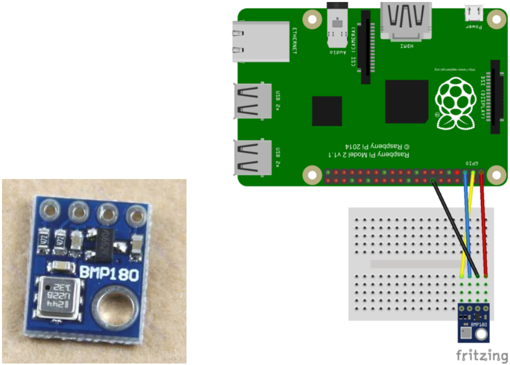
```
'''
    Read Gyro and Accelerometer by Interfacing Raspberry Pi with MPU6050 using Python
    http://www.electronicwings.com
'''
import smbus             #import SMBus module of I2C
from time import sleep   #import

#some MPU6050 Registers and their Address
PWR_MGMT_1   = 0x6B
SMPLRT_DIV   = 0x19
CONFIG       = 0x1A
GYRO_CONFIG  = 0x1B
INT_ENABLE   = 0x38
ACCEL_XOUT_H = 0x3B
ACCEL_YOUT_H = 0x3D
ACCEL_ZOUT_H = 0x3F
GYRO_XOUT_H  = 0x43
GYRO_YOUT_H  = 0x45
GYRO_ZOUT_H  = 0x47
```

Sample code (continued)

```
def MPU_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_raw_data(addr):
    #Accelero 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)

        #to get signed value from mpu6050
        if(value > 32768):
                value = value - 65536
        return value
```

Sample code (continued)

```
bus = smbus.SMBus(1)    # or bus = smbus.SMBus(0) for older version boards
Device_Address = 0x68   # MPU6050 device address

MPU_Init()

print (" Reading Data of Gyroscope and Accelerometer")

while True:

    #Read Accelerometer raw value
    acc_x = read_raw_data(ACCEL_XOUT_H)
    acc_y = read_raw_data(ACCEL_YOUT_H)
    acc_z = read_raw_data(ACCEL_ZOUT_H)

    #Read Gyroscope raw value
    gyro_x = read_raw_data(GYRO_XOUT_H)
    gyro_y = read_raw_data(GYRO_YOUT_H)
    gyro_z = read_raw_data(GYRO_ZOUT_H)

    #Full scale range +/- 250 degree/C as per sensitivity scale factor
    Ax = acc_x/16384.0
    Ay = acc_y/16384.0
    Az = acc_z/16384.0

    Gx = gyro_x/131.0
    Gy = gyro_y/131.0
    Gz = gyro_z/131.0

    print ("Gx=%.2f" %Gx, "\tGy=%.2f" %Gy, "\tGz=%.2f" %Gz, \
           "\tAx=%.2f g" %Ax, "\tAy=%.2f g" %Ay, "\tAz=%.2f g" %Az)
    sleep(1)
```



(a) Sensor (b) Wiring

Figure 10: BMP180 altitude, temperature & pressure sensor.

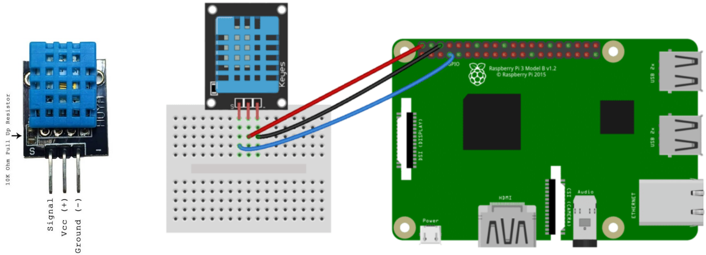
https://thepihut.com/blogs/raspberry-pi-tutorials/18025084-sensors-pressure-temperature-and-altitude-with-the-bmp180

Sample code

```
#!/usr/bin/python

import Adafruit_BMP.BMP085 as BMP085 # Imports the BMP library

# Create an 'object' containing the BMP180 data
sensor = BMP085.BMP085()

print 'Temp = {0:0.2f} *C'.format(sensor.read_temperature())                           # Temperature in Celcius
print 'Pressure = {0:0.2f} Pa'.format(sensor.read_pressure())                          # The local pressure
print 'Altitude = {0:0.2f} m'.format(sensor.read_altitude())                           # The current altitude
print 'Sealevel Pressure = {0:0.2f} Pa'.format(sensor.read_sealevel_pressure()) # The sea-level pressure
```

Raspberry Pi based Data Loggers

Humidity & Temperature Sensor



(a) Sensor (b) Wiring

Figure 11: DHT11 humidity & temperature Sensor.

http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-the-raspberry-pi/

Sample code

```
#!/usr/bin/python
import sys
import Adafruit_DHT

while True:
    humidity, temperature = Adafruit_DHT.read_retry(11, 4)
    print 'Temp: {0:0.1f} C  Humidity: {1:0.1f} %'.format(temperature, humidity)
```

Sense HAT is “Jack of many trades” and comes with:

- a magnetometer
- a gyroscope (sensing pitch, roll, and yaw) and an accelerometer,
- sensors for temperature, humidity, and barometric pressure
- a joystick and an 8×8 LED matrix

(a) Sense HAT (b) Sense HAT & RPi (c) Piggyback

Figure 12: Sense HAT.

https://www.raspberrypi.org/products/sense-hat/

Raspberry Pi based Data Loggers

Soil Moisture Sensor for Smart Gardening

- **Note:** This data logger needs relay.

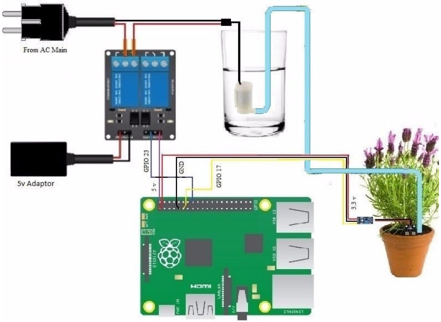


Figure 13: Smart gardening using 5V 10A 2 channel relay to power the water pump.

https://www.hackster.io/mtechkiran/smart-home-gardening-system-using-raspberry-pi-1570a7

- **Note:** This data logger needs ADC (MCP3204).

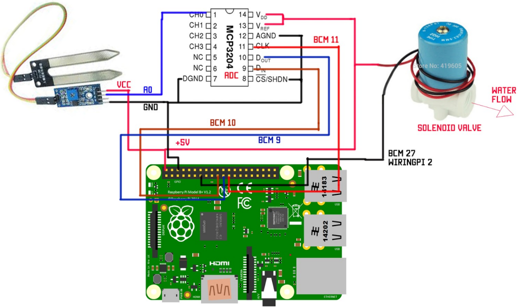
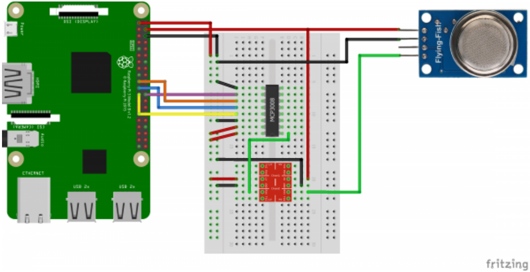


Figure 14: Smart irrigation project making use of ADC and solenoid valve.

http://www.embeddedstudy.com/2017/09/iot-based-smart-irrigation-project-on.html

- **Note:** This data logger needs ADC (MCP3008).

(a) Sensor



(b) Wiring

Figure 15: MQ-2 (LPG, methane, butane, smoke) gas sensor.

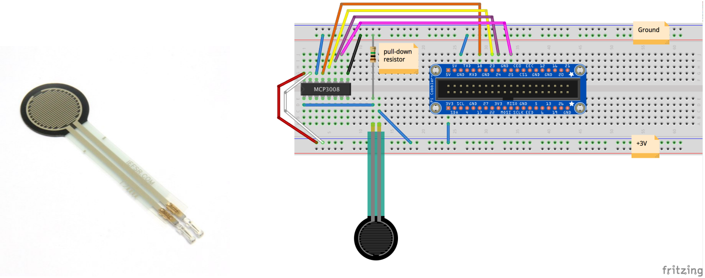
https://tutorials-raspberrypi.com/configure-and-read-out-the-raspberry-pi-gas-sensor-mq-x/

Source codes: `git clone https://github.com/tutRPi/Raspberry-Pi-Gas-Sensor-MQ`

Table 1: Gas sensors

Sensor	Gas(es)
MQ-2	Methane, Butane, LPG, smoke
MQ-3	Alcohol, Ethanol, smoke
MQ-4	Methane, CNG Gas
MQ-5	Natural gas, LPG
MQ-6	LPG, butane gas
MQ-7	Carbon Monoxide
MQ-8	Hydrogen Gas
MQ-9	Carbon Monoxide, flammable gasses
MQ-131	Ozone
MQ-135	Benzene, Alcohol, smoke
MQ-136	Hydrogen Sulfide gas
MQ-137	Ammonia
MQ-138	Benzene, Toluene, Alcohol, Acetone, Propane, Formaldehyde gas
MQ-214	Methane, Natural gas
MQ-216	Natural gas, Coal gas
MQ-303A	Alcohol, Ethanol, smoke
MQ-306A	LPG, butane gas
MQ-307A	Carbon Monoxide

- **Note:** This data logger needs ADC (MCP3008).

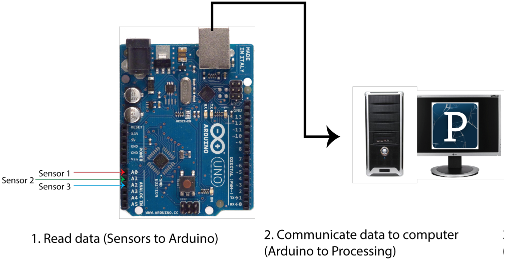


(a) Sensor (b) Wiring

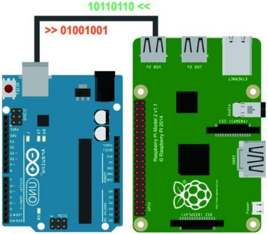
Figure 16: Connecting the force sensitive resistor and ADC to RPi's GPIO

http://arduinolearning.com/code/force-sensitive-resistor-example.php
http://acaird.github.io/computers/2015/01/07/raspberry-pi-fsr

“Raspberry Pi + Arduino”
based
Data Loggers



(a) Arduino + PC



(b) Arduino + RPi

Figure 17: Connecting Arduino microcontroller using USB ports.

http://www.hackerscapes.com/2014/11/how-to-save-data-from-arduino-to-a-csv-file-using-processing/

- **Android Things:** (codenamed Brillo) is an Android-based embedded operating system platform by Google, announced at Google I/O 2015. It is aimed to be used with low-power and memory constrained Internet of Things (IoT) devices, which are usually built from different MCU platforms.

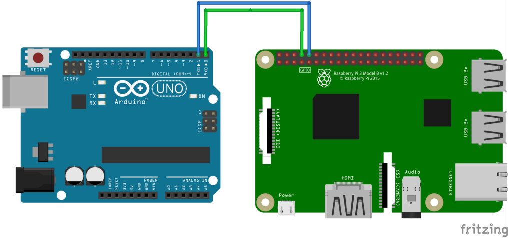


Figure 18: Connecting RPi3 with Android Things to Arduino.

https://en.wikipedia.org/wiki/Android_Things
https://medium.com/@bastermark3/connecting-raspberry-pi-3-with-android-things-to-arduino-51d202006379

- **I2C:** is a useful bus that allows data exchange between microcontrollers and peripherals with a minimum of wiring.

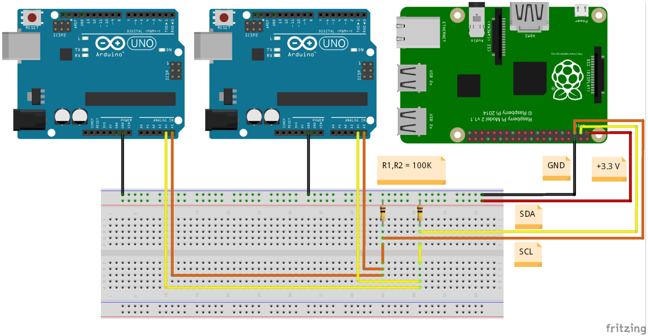


Figure 19: Use a Raspberry as Master for I2C Bus and Arduino Uno as Slave (127 slaves capability).

https://learn.sparkfun.com/tutorials/raspberry-pi-spi-and-i2c-tutorial
http://fritzing.org/projects/i2c-raspberrypi2-master-to-arduino-slave

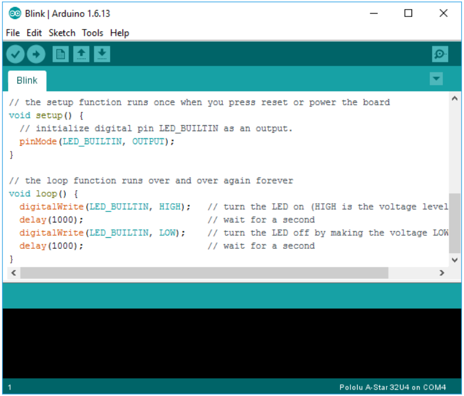
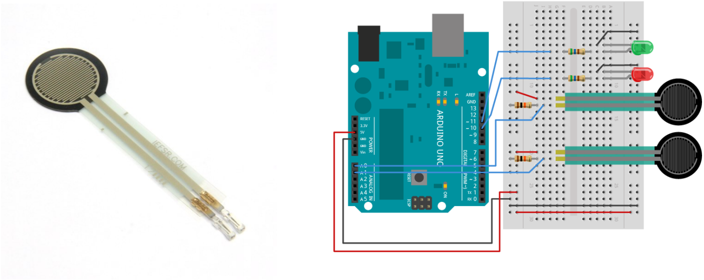


Figure 20: Arduino IDE.

https://www.arduino.cc/en/Main/Software?

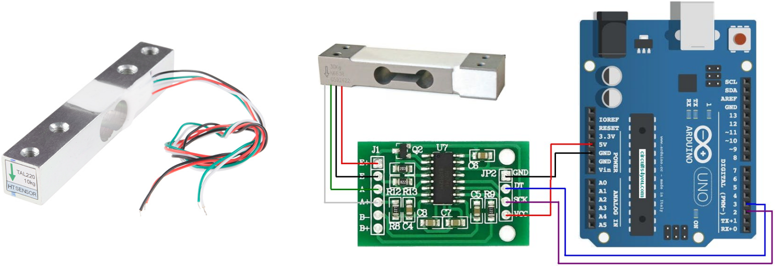


(a) Sensor (b) Wiring

Figure 21: Using Arduino to read force-sensing resistors.

http://arduinolearning.com/code/force-sensitive-resistor-example.php

https://itp.nyu.edu/archive/physcomp-spring2014/Labs/AnalogIn



(a) Sensor (b) Wiring

Figure 22: Load cell connections to HX711 load cell amplifier module and Arduino.

https://www.sparkfun.com/products/13329
https://www.hackster.io/MOHAN_CHANDALURU/hx711-load-cell-amplifier-interface-with-arduino-fa47f3

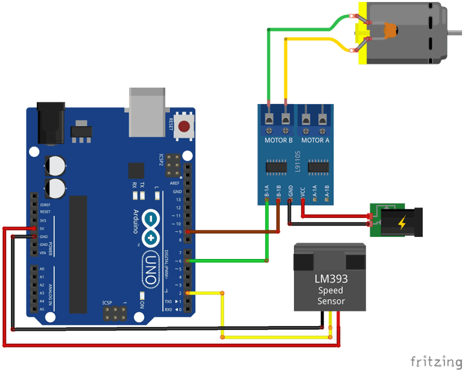


Figure 23: Connecting an infrared speed sensor based on the LM393 chip.

https://www.brainy-bits.com/speed-sensor-with-arduino/

Sample code

```
#include "TimerOne.h"
unsigned int counter=0;

int b1a = 6;  // L9110 B-1A
int b1b = 9;  // L9110 B-1B

void docount()  // counts from the speed sensor
{
  counter++;    // increase +1 the counter value
}

void timerIsr()
{
  Timer1.detachInterrupt();             // stop the timer
  Serial.print("Motor Speed: ");
  int rotation = (counter / 20);        // divide by number of holes in Disc
  Serial.print(rotation,DEC);
  Serial.println(" Rotation per seconds");
  counter=0;  //  reset counter to zero
  Timer1.attachInterrupt( timerIsr );  // enable the timer
}

void setup()
{
  Serial.begin(9600);

 pinMode(b1a, OUTPUT);
 pinMode(b1b, OUTPUT);

  Timer1.initialize(1000000);          // set timer for 1sec
  attachInterrupt(0, docount, RISING); // increase counter when speed sensor pin goes High
  Timer1.attachInterrupt( timerIsr );  // enable the timer
}
```

Sample code (continued)

```
void loop()
{
  int potvalue = analogRead(1);  // Potentiometer connected to Pin A1
  int motorspeed = map(potvalue, 0, 680, 255, 0);
  analogWrite(b1a, motorspeed);  // set speed of motor (0-255)
  digitalWrite(b1b, 1);          // set rotation of motor to Clockwise
}
```

Bibliography

1. HEITZ, R. (2016): *Hello Raspberry Pi!*, Manning (ISBN: 9781617292453)
2. RICHARDSON, M. & WALLACE, S. (2013): *Getting Started with Raspberry Pi*, O'Reilly (ISBN: 978-1-449-34421-4)
3. SOPER, M. E. (2017): *Expanding Your Raspberry Pi*, Apress (ISBN: 978-1-4842-2922-4)

. . . must end

- . . . and I end my presentation with two supplications

رَّبِّ زِدْنِي عِلْمًا

my Lord! increase me in knowledge

(Taa-Haa (20):114)

اَللّٰهُمَّ اِنَّا نَسْئَلُكَ عِلْمًا نَافِعًا

O Allah! We ask You for knowledge that is of benefit

(ibn Majah)