ENTHU TECHNOLOGY SOLUTIONS
INDIA PVT LTD
Training | Support | Development

ETS IoT Trainer Kit

A platform to Experiment Internet of Things

User Guide

*Email **info@enthutech.in** for technical / sales support*

Note 1:

ETS IoT KIT user guide is documented based on our in house developer experience. User can refer the examples / references provided in this document for only demonstration purposes and the same example should not be placed in any of their product code. And the third party library credit goes to the respective organization/developers.

Note 2:

Different text formats have been used throughout this document.

- ***Warning format:*** user should give attention to this kind of format as it is very critical for device durability.
- *Software Example format*: Example applications developed in Python 2.x
- *LxTerminal command format* : commands which has been used in LxTerminal

Warning:

The device can be impaired or damaged if the power source is inadequate.

- ***Use only certified USB power adapters/connectors.***
- ***Don't power up using Laptop/PC***

This product, like all microcontroller products, uses semiconductors that can be damaged by electrostatic discharge (ESD). We recommend to use ESD protection while you are using ETS IoT KIT. Damage due to inappropriate handling is not covered by the warranty.

Contents

REFERENCES

- http://www.raspberrypi.org/downloads
- https://www.bosch-sensortec.com/bst/products/all_products/bme280
- https://www.bosch-sensortec.com/bst/products/all_products/bmi160
- https://learn.adafruit.com/mcp230xx-gpio-expander-on-the-raspberry-pi/using-the-library
- http://ww1.microchip.com/downloads/en/DeviceDoc/21952b.pdf
- https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf
- https://people.csail.mit.edu/albert/bluez-intro/
- http://people.csail.mit.edu/rudolph/Teaching/Articles/BTBook.pdf
- http://wiringpi.com/
- http://www.raspberry-projects.com/pi/programming-in-c/getting-your-raspberry-pi-ready-for-c-programming
- https://www.python.org/
- https://www.raspberrypi.org/documentation/linux/software/python.md
- https://www.raspberrypi.org/community/
- https://raspberrypi.stackexchange.com/
- *https://github.com/pimoroni/adxl345-python*

GETTING STARTED

ETS IoT kit is an all-in-one prototyping platform for sensor based IoT projects. It's packed with state of the art sensor technology and ready-to-use software applications, capable of fulfilling all your IoT application needs.

It contains inbuilt raspberry pi3 with all in one sensor platform to enable infinite IoT application prototype possibilities.

Software Download and Installation

The software the Raspberry pi is available online. You can download the software from http://www.raspberrypi.org/downloads/noobs/ and copy the OS in your SD card and insert in into your Raspberry pi board.

HARDWARE SPECIFICATION

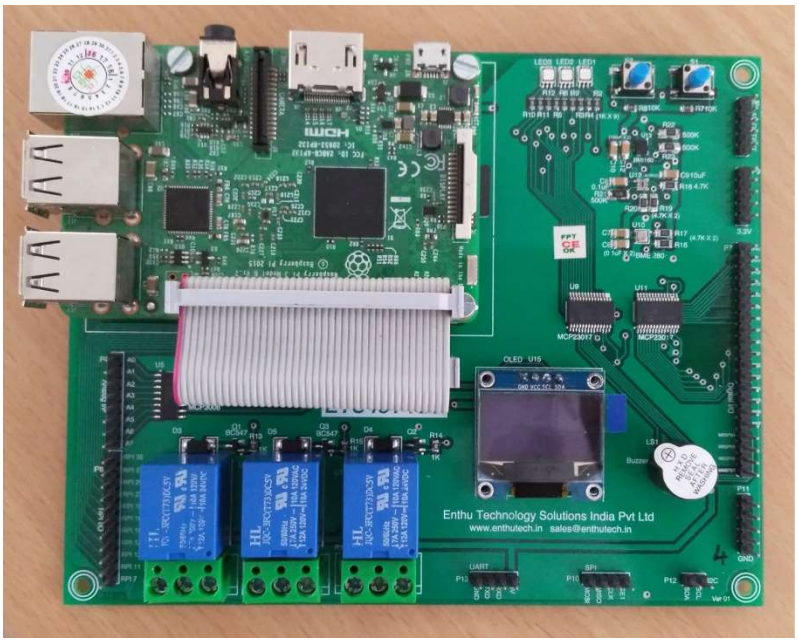


Figure: Top view of ETS IoT KIT

Hardware Specifications:

- SoC: Broadcom BCM2837
- CPU: 4× ARM Cortex-A53, 1.2GHz
- GPU: Broadcom VideoCore IV
- RAM: 1GB LPDDR2 (900 MHz)
- Communication Interfaces
    - Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless
    - Bluetooth 4.1 Classic, Bluetooth Low Energy
- Inertial Sensors
    - Accelerometer
    - Gyroscope
- Environmental sensor
    - Temperature
    - Humidity
    - Pressure
- User interfaces
    - 3x3 channel relay
    - 3 RGB LED
    - 2 Push button
    - Buzzer
    - OLED display
- Ports
    - UART
    - SPI
    - I2C
    - 8 Analog I/O
    - 31 GPIOs to interface external components
    - 4 USB 2.0 port
    - Micro SD card 1
    - 3.5mm analog audio-video jack
    - 5x5V port
    - 3x3.3V port
    - HDMI
    - Camera Serial Interface(CSI)
    - Display Serial Interface(DSI)

SETTING UP ETS IOT KIT

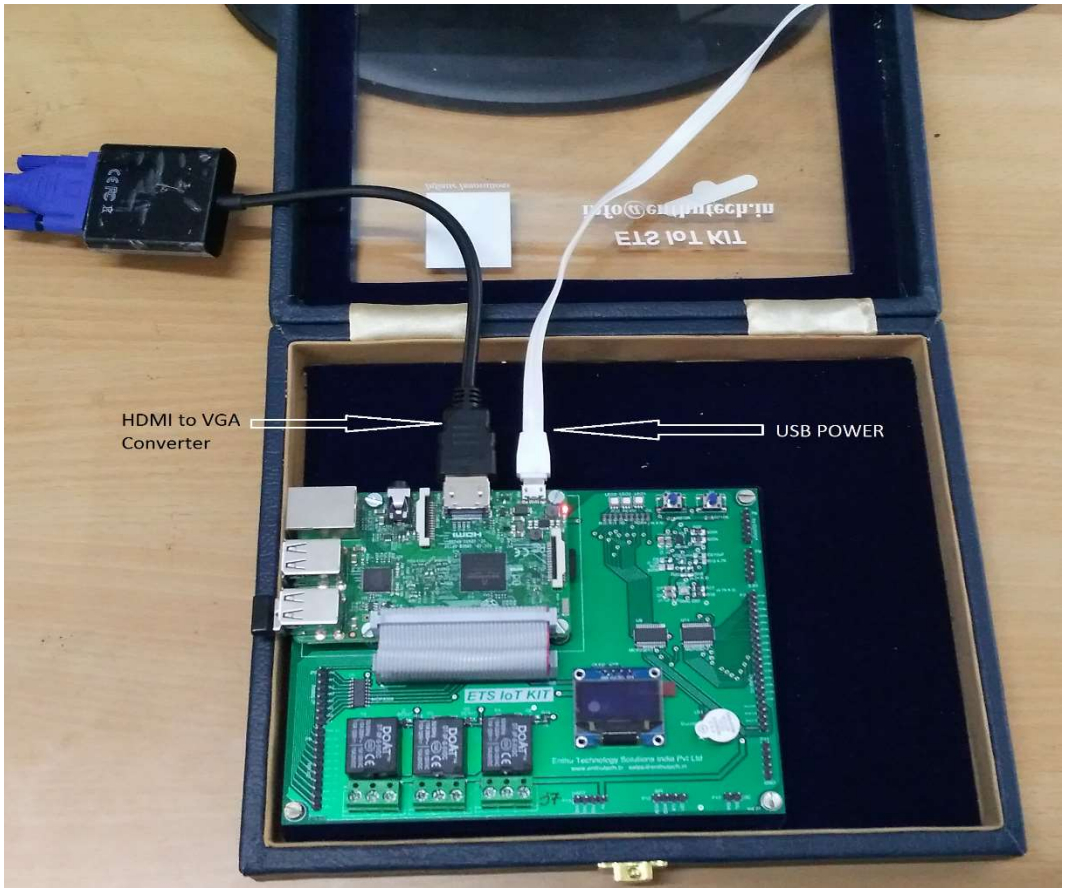


Figure: Power / Monitor connection of ETS IoT KIT

Connect your ETS IoT KIT with monitor using HDMI to VGA converter cable and Power adaptor. Switch ON the power supply and it will start working. Once turn ON the power adaptor you will see the below image.

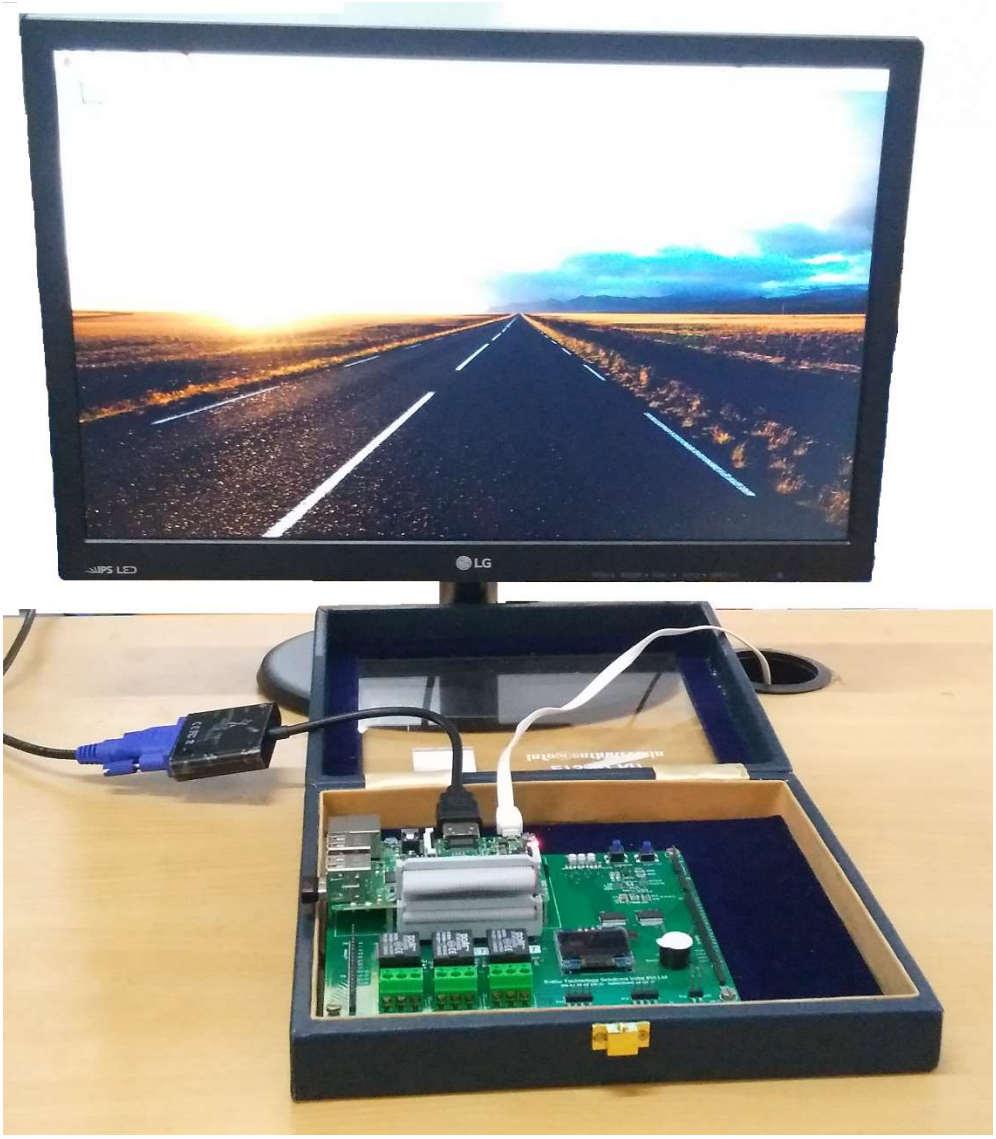


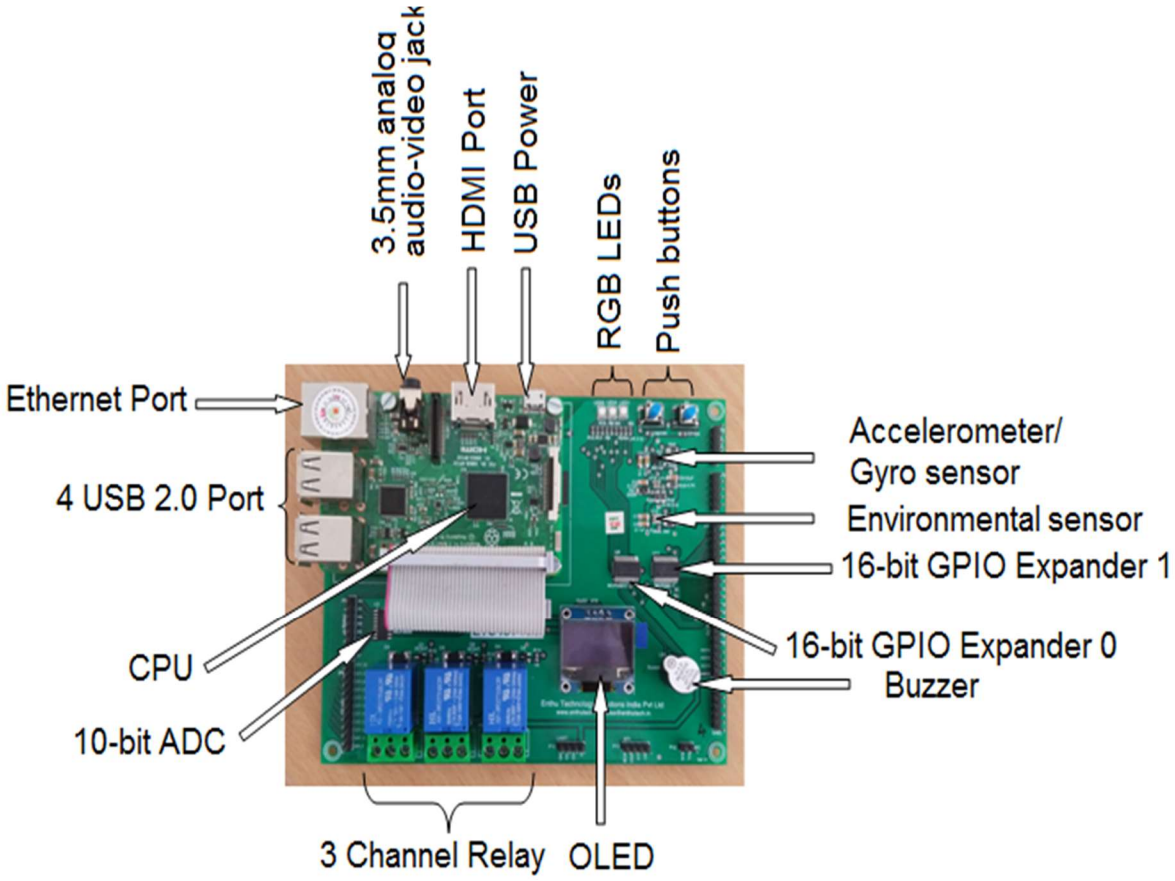
Figure: Full view of ETS IoT KIT with monitor

FUNCTIONAL DESCRIPTION

Block Diagram

The following Picture 2 shows a simplified block diagram of the IoT Kit:

Figure: Block diagram of IoT Kit



Sensors

BMI160: Inertial Measurement Unit

The BMI160 is a small, low-power, low-noise 16-bit inertial measurement unit designed for use in mobile applications like augmented reality or indoor navigation which require highly accurate, real time sensor data. In full operation mode, with both the accelerometer and the gyroscope enabled, the current consumption is typically 950 µA, enabling always-on applications in battery driven devices.

Table: BMI160 Key Features

Specification	Value
Digital Resolution	Accelerometer (A): 16 bit Gyroscope (G): 16 bit
Measurement Ranges (programmable)	(A): ± 2 g, ± 4 g, ± 8 g, ± 16 g (G): ± 125 °/s, ± 250 °/s, ± 500 °/s, ± 1000 °/s, ± 2000 °/s
Sensitivity (calibrated)	(A): ±2g: 16384 LSB/g ±4g: 8192 LSB/g ±8g: 4096 LSB/g ±16g: 2048 LSB/g (G): ±125 °/s: 262.4 LSB/ °/s ±250°/s: 131.2 LSB/ °/s ±500 °/s: 65.6 LSB/ °/s ±1000 °/s: 32.8 LSB/ °/s ±2000 °/s: 16.4 LSB/ °/s
Zero-Point Offset	(A): ±40mg (G): ± 10 °/s
Noise Density (typical)	(A): 180 µg/√Hz (G): 0.008 °/s/√Hz
Bandwidths (programmable)	1600 Hz … 25/32 Hz

For more information please refer to the datasheet:

http://www.mouser.com/ds/2/783/BST-BMI160-DS000-07-786474.pdf

BME280: Environmental Sensor

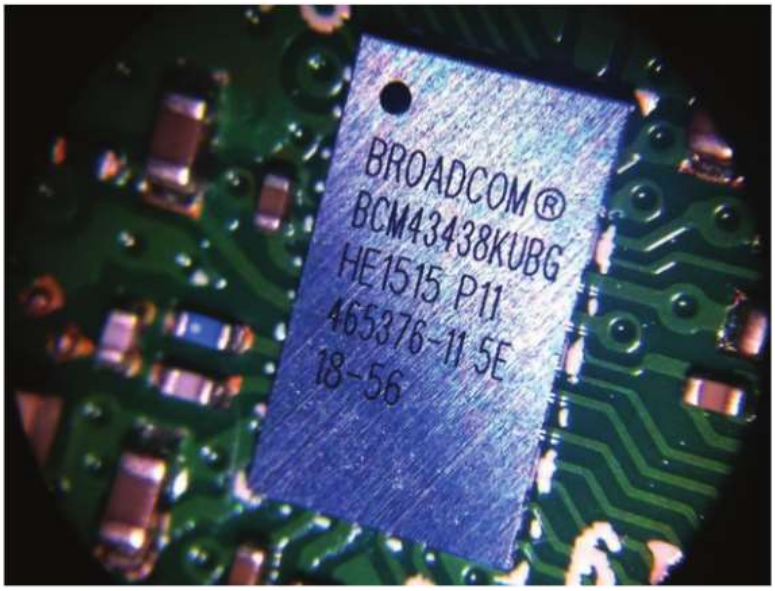
The BME280 is an integrated environmental sensor developed specifically for mobile applications where size and low power consumption are key design constraints. The unit combines individual high-linearity, high-accuracy sensors for pressure, humidity and temperature, designed for low current consumption (3.6 µA @ 1 Hz), and long-term stability. The humidity sensor features an extremely fast response time which supports performance requirements for emerging applications such as context awareness, and high accuracy over a wide temperature range. The pressure sensor is an absolute barometric pressure sensor which features exceptionally high accuracy and resolution at very low noise. The integrated temperature sensor has been optimized for very low noise and high resolution. It is primarily used for temperature compensation of the pressure and humidity sensors, and can also be used for estimating ambient temperature.

Table: Sensor Measurement Ranges and Accuracies

Parameter	Condition	Min		Max	Unit
Operating Temperature Range	Operational	-20	25	+60	°C
	Full Accuracy	0		+60	°C
Operating Pressure Range		300		1100	hPa
Absolute Accuracy Pressure			±10		hPa
Absolute Accuracy Temperature			±2		K
Absolute Accuracy Humidity			±10		%RH
Average Current Consumption (1Hz Data Refresh Rate)	H, T		1.8		µA
	P, T		2.8		µA
	H, P, T		3.6		µA

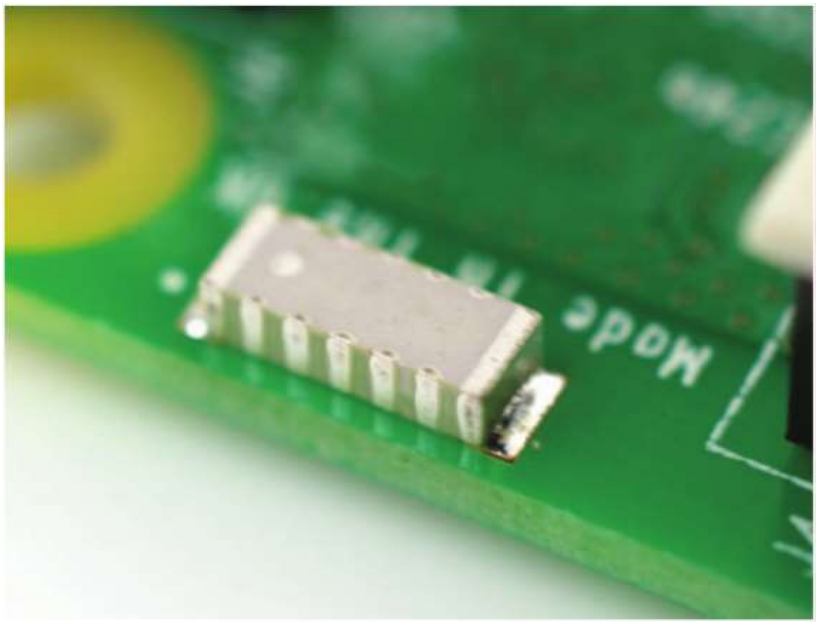
Wireless Radio

Inbuilt wireless radio is very small, its markings can only be properly seen through a microscope or magnifying glass, the Broadcom BCM43438 chip provides 2.4GHz 802.11n wireless LAN, Bluetooth Low Energy, and Bluetooth 4.1 Classic radio support. Cleverly built directly onto the board to keep costs down, rather than the more common fully qualified module approach, its only unused feature is a disconnected FM radio receiver.



Antenna

There's no need to connect an external antenna to the Raspberry Pi 3. Radios are connected to this chip antenna soldered directly to the board, to keep the size of the device to a minimum. Despite its diminutive stature, this antenna should be more than capable of picking up wireless LAN and Bluetooth signals – even through walls.



SD Card

The Raspberry Pi should work with any compatible SD card, although there are some guidelines that should be followed:

Memory Size:

For installation of NOOBS, the minimum recommended card size is 8GB. For image installations, we recommend a minimum of 4GB. Some distributions, specifically LibreELEC and Arch, can run on much smaller cards.

Card Class:

The card class determines the sustained write speed for the card; a class 4 card will be able to write at 4MB/s, whereas a class 10 should be able to attain 10 MB/s. However, it should be noted that this does not mean a class 10 card will outperform a class 4 card for general usage, because often this write speed is achieved at the cost of read speed and increased seek times.

I2C Devices

It has six i2c devices.

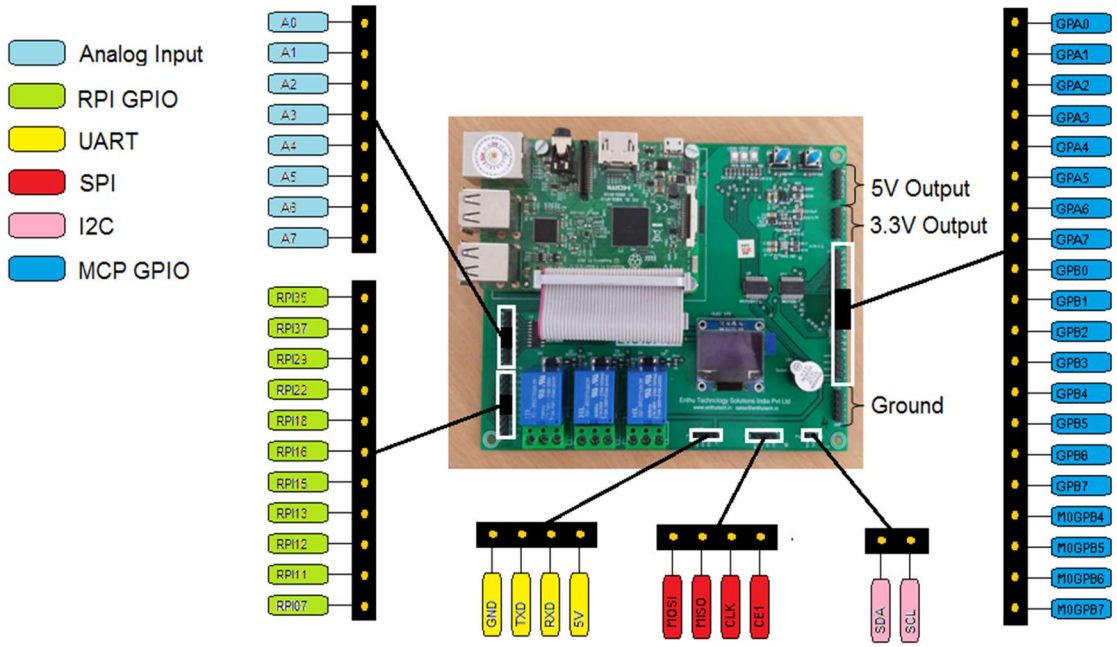
- MCP23017-IC1
- MCP23017-IC2
- BME280
- BMI160
- OLED

SPI Devices

It has an IC called MCP3008 10Bit ADC converter connected with Chip Enable 0 (CE0).

PIN CONFIGURATION

Below diagram describes PIN configuration details of ETS IoT KIT,



ETS IoT KIT consists of two MCP23017 and one MCP3008 which are used for I/O expander and On board component connectivity. Below tables describes PIN mapping details of the same.

MCP23017 – U9

MCP23017-U9	User Interface
GPA0	LED1_RED
GPA1	LED1 _GREEN
GPA2	LED1_BLUE
GPA3	LED2_RED
GPA4	LED2_GREEN
GPA5	LED2_BLUE
GPA6	LED3_RED
GPA7	LED3_GREEN
GPB0	LED3_BLUE

GPB1	SWITCH1
GPB2	SWITCH2
GPB3	BUZZER
GPB4	M0GPB4
GPB5	M0GPB5
GPB6	M0GPB6
GPB7	M0GPB7

MCP23017 – U11

MCP23017-U11	User Interface
GPA0	GPA0
GPA1	GPA1
GPA2	GPA2
GPA3	GPA3
GPA4	GPA4
GPA5	GPA5
GPA6	GPA6
GPA7	GPA7
GPB0	GPB0
GPB1	GPB1
GPB2	GPB2
GPB3	GPB3
GPB4	GPB4
GPB5	GPB5
GPB6	GPB6
GPB7	GPB7

MCP3008-U5

MCP3008	User Interface
CH0	A0
CH1	A1
CH2	A2
CH3	A3
CH4	A4
CH5	A5
CH6	A6
CH7	A7

APPLICATION EXAMPLES

This section gives detailed connection diagrams and application coding example for each on board components. All demo programs are placed in following folder path of each ETS IoT KIT

Home/pi/Desktop/ETS IoT KIT Demo/DemoCode

Note: All example code is based on python 2.7 and Adafruit library

LED:

ETS IoT KIT has 3 RGB LEDs which has been connected to raspberry pi through MCP23017. All demo programs are used Adafruit library.

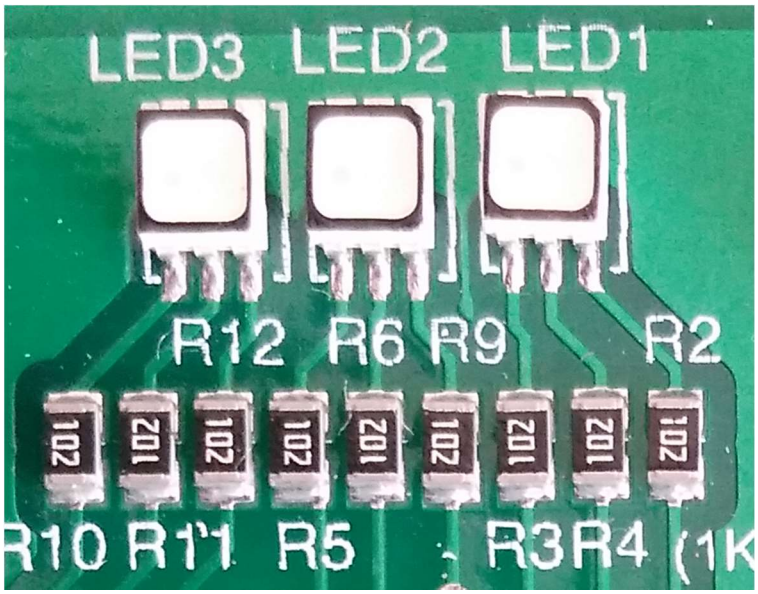


Figure: RGB LED Image in ETS-IoT Trainer Kit

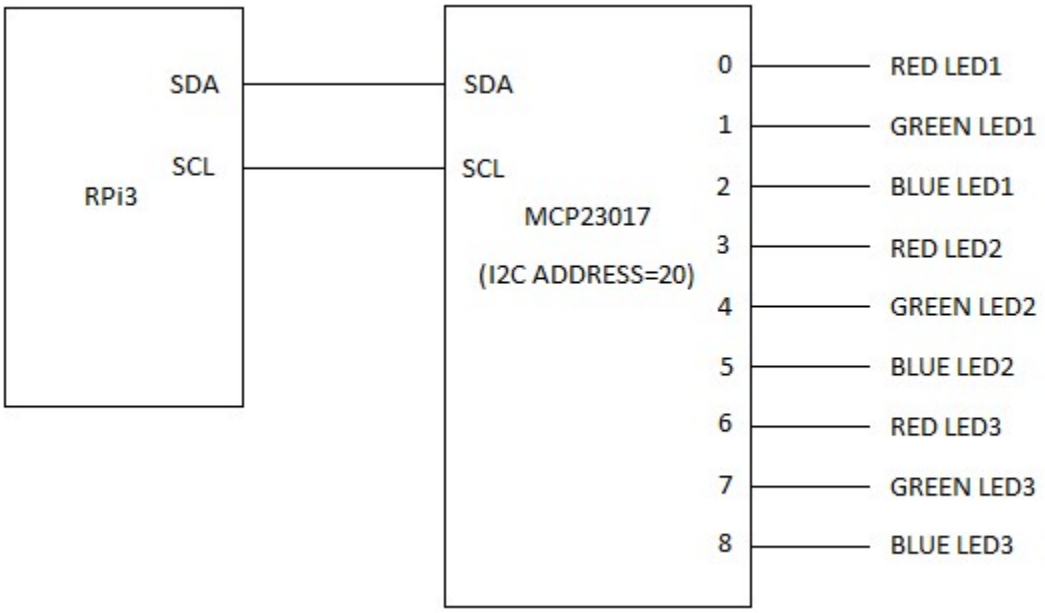


Figure: Connection diagram of RGB LED

Software Example:

```
# Application to blink all RED LEDs from 3 RGB LEDs

#mcp23017 library path
import time

import sys
sys.path.append('/home/pi/Adafruit-Raspberry-Pi-Python-Code-legacy/Adafruit_MCP230xx')
from Adafruit_MCP230XX import Adafruit_MCP230XX

#mcp IC configuration
mcp = Adafruit_MCP230XX(busnum = 1, address = 0x20, num_gpios = 16)

#mcp input/output configuration
mcp.config(0, mcp.OUTPUT)
mcp.config(1, mcp.OUTPUT)
mcp.config(2, mcp.OUTPUT)
mcp.config(3, mcp.OUTPUT)
mcp.config(4, mcp.OUTPUT)
mcp.config(5, mcp.OUTPUT)
mcp.config(6, mcp.OUTPUT)
mcp.config(7, mcp.OUTPUT)
```

```
mcp.config(8, mcp.OUTPUT)


while True:
   #RGB LED blink
   mcp.output(0, 1) #RED LED1 ON
   mcp.output(3, 1) #RED LED2 ON
   mcp.output(6, 1) #RED LED3 ON
   time.sleep(1)
   mcp.output(0, 0) #RED LED1 OFF
   mcp.output(3, 0) #RED LED1 OFF
   mcp.output(6, 0) #RED LED1 OFF
   time.sleep(1)
```

SWITCH:

ETS IoT KIT has 2 PUSH BUTTON which has been connected to raspberry pi through MCP23017.

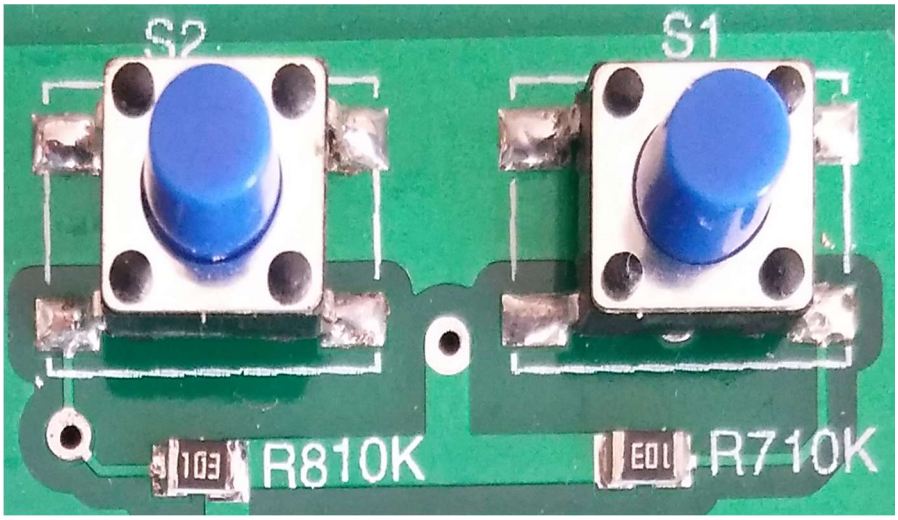


Figure: SWITCH Image in ETS-IoT Trainer Kit

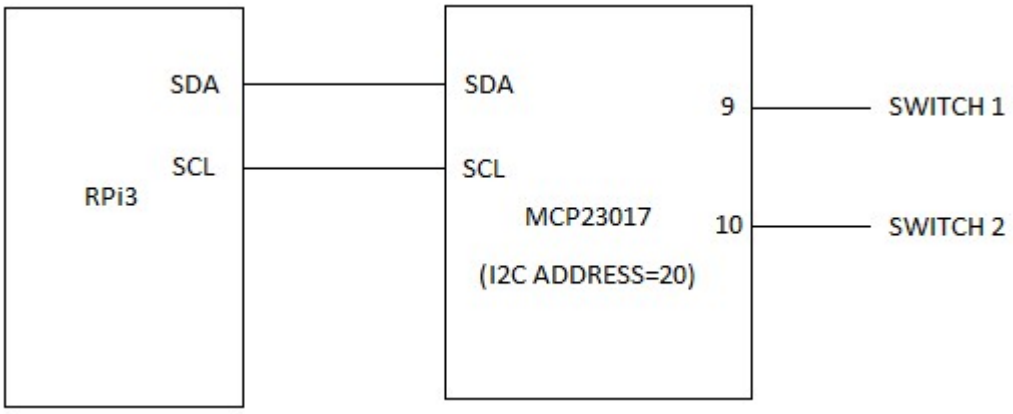


Figure: Connection diagram of SWITCH

Software Example:

```
#Application to read and print the status of both PUSH BUTTON.

import time
import sys

sys.path.append('/home/pi/Adafruit-Raspberry-Pi-Python-Code-legacy/Adafruit_MCP230xx')

from Adafruit_MCP230XX import Adafruit_MCP230XX

mcp = Adafruit_MCP230XX(busnum = 1, address = 0x20, num_gpios = 16)

# Set pin 3 to input with the pullup resistor enabled
mcp.config(9, mcp.INPUT)
mcp.pullup(9, 1)
mcp.config(10, mcp.INPUT)
mcp.pullup(10, 1)

while (True):
   print "Pin 9 = %d" % (mcp.input(9))
   print "Pin 10 = %d" % (mcp.input(10))
   time.sleep(2)
```

BUZZER:

ETS IoT KIT has a BUZZER which has been connected to raspberry pi through MCP23017. All demo programs are used Adafruit library.

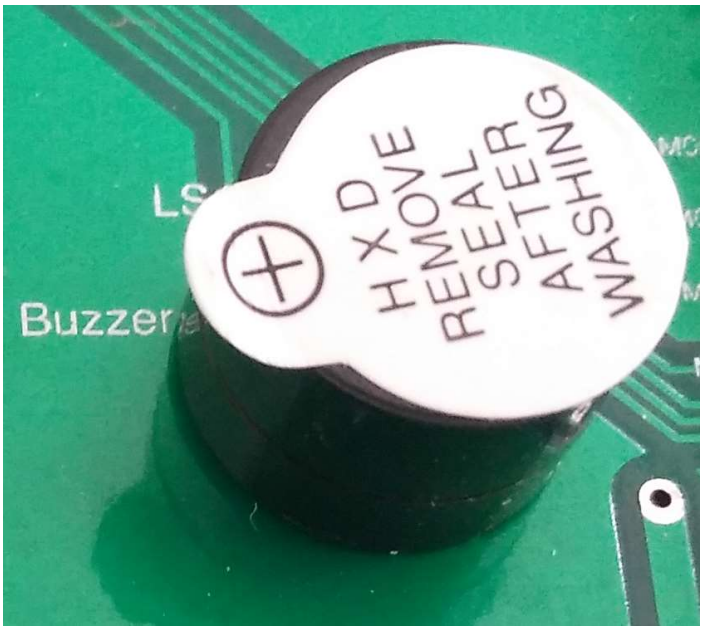


Figure: BUZZER Image in ETS-IoT Trainer Kit

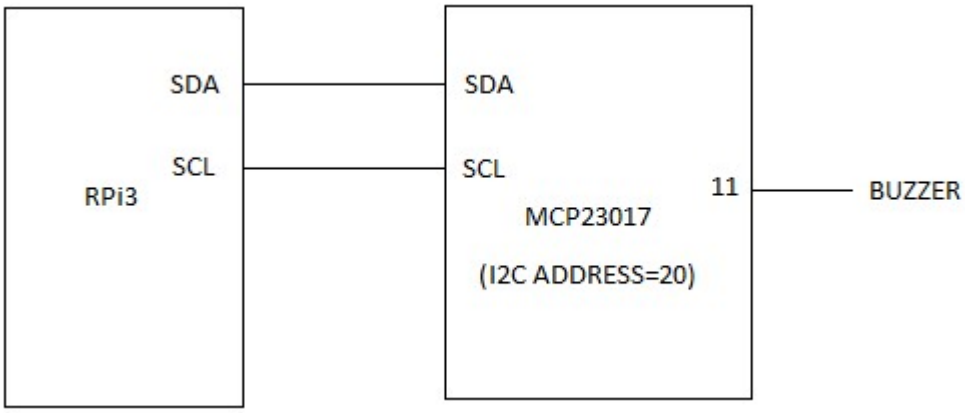


Figure: Connection diagram of BUZZER

Software Example:

```
# Application to ON/OFF the BUZZER
import time
import sys
sys.path.append('/home/pi/Adafruit-Raspberry-Pi-Python-Code-legacy/Adafruit_MCP230xx')

from Adafruit_MCP230XX import Adafruit_MCP230XX

mcp = Adafruit_MCP230XX(busnum = 1, address = 0x20, num_gpios = 16)

#mcp input/output configuration
mcp.config(11, mcp.OUTPUT)

while (True):
   mcp.output(11, 1) #BUZZER ON
   time.sleep(1)
   mcp.output(11, 0) #BUZZER OFF
   time.sleep(1)
```

BME280

ETS IoT KIT has an ENVIRONMENTAL Sensor which has been connected to raspberry pi through I2C. It will show the value of TEMPERATURE, PRESSURE and HUMIDITY. The I2C address of BME280 sensor in ETS IoT KIT is 0x76.

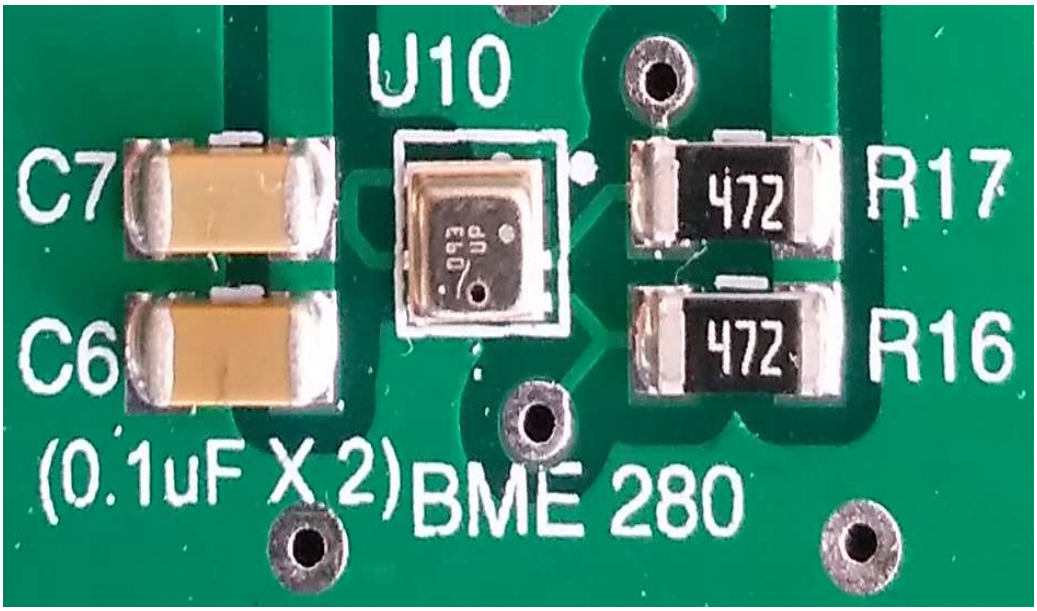


Figure: BME280 Image in ETS-IoT Trainer Kit

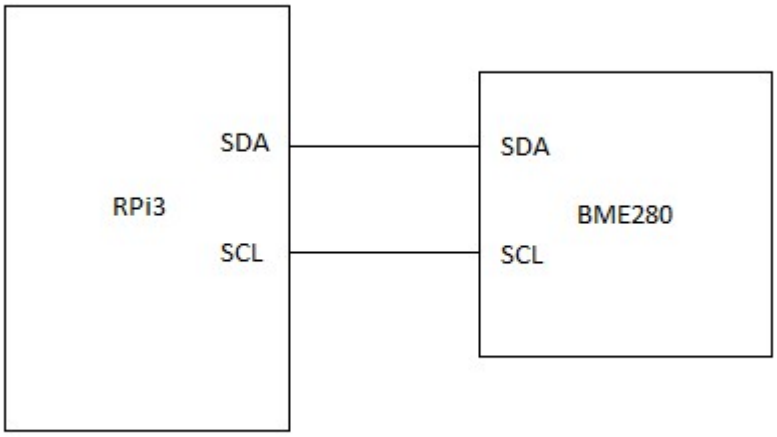


Figure: Connection diagram of BME280

Software Example:

```
# Application read and display Temperature (C), Pressure (hPa), Humidity(%)
import smbus
import time
from ctypes import c_short
from ctypes import c_byte
from ctypes import c_ubyte
DEVICE = 0x76 # Default device I2C address

bus = smbus.SMBus(1) # Rev 2 Pi, Pi 2 & Pi 3 uses bus 1
                     # Rev 1 Pi uses bus 0

def getShort(data, index):
  # return two bytes from data as a signed 16-bit value
  return c_short((data[index+1] << 8) + data[index]).value

def getUShort(data, index):
  # return two bytes from data as an unsigned 16-bit value
  return (data[index+1] << 8) + data[index]

def getChar(data,index):
  # return one byte from data as a signed char
  result = data[index]
  if result > 127:
    result -= 256
  return result

def getUChar(data,index):
  # return one byte from data as an unsigned char
  result =  data[index] & 0xFF
  return result

def readBME280ID(addr=DEVICE):
  # Chip ID Register Address
  REG_ID     = 0xD0
  (chip_id, chip_version) = bus.read_i2c_block_data(addr, REG_ID, 2)
  return (chip_id, chip_version)

def readBME280All(addr=DEVICE):
  # Register Addresses
  REG_DATA = 0xF7
  REG_CONTROL = 0xF4
  REG_CONFIG  = 0xF5
```

```
REG_CONTROL_HUM = 0xF2
REG_HUM_MSB = 0xFD
REG_HUM_LSB = 0xFE

# Oversample setting - page 27
OVERSAMPLE_TEMP = 2
OVERSAMPLE_PRES = 2
MODE = 1

# Oversample setting for humidity register - page 26
OVERSAMPLE_HUM = 2
bus.write_byte_data(addr, REG_CONTROL_HUM, OVERSAMPLE_HUM)

control = OVERSAMPLE_TEMP<<5 | OVERSAMPLE_PRES<<2 | MODE
bus.write_byte_data(addr, REG_CONTROL, control)

# Read blocks of calibration data from EEPROM
# See Page 22 data sheet
cal1 = bus.read_i2c_block_data(addr, 0x88, 24)
cal2 = bus.read_i2c_block_data(addr, 0xA1, 1)
cal3 = bus.read_i2c_block_data(addr, 0xE1, 7)

# Convert byte data to word values
dig_T1 = getUShort(cal1, 0)
dig_T2 = getShort(cal1, 2)
dig_T3 = getShort(cal1, 4)

dig_P1 = getUShort(cal1, 6)
dig_P2 = getShort(cal1, 8)
dig_P3 = getShort(cal1, 10)
dig_P4 = getShort(cal1, 12)
dig_P5 = getShort(cal1, 14)
dig_P6 = getShort(cal1, 16)
dig_P7 = getShort(cal1, 18)
dig_P8 = getShort(cal1, 20)
dig_P9 = getShort(cal1, 22)

dig_H1 = getUChar(cal2, 0)
dig_H2 = getShort(cal3, 0)
dig_H3 = getUChar(cal3, 2)

dig_H4 = getChar(cal3, 3)
dig_H4 = (dig_H4 << 24) >> 20
```

```
dig_H4 = dig_H4 | (getChar(cal3, 4) & 0x0F)

dig_H5 = getChar(cal3, 5)
dig_H5 = (dig_H5 << 24) >> 20
dig_H5 = dig_H5 | (getUChar(cal3, 4) >> 4 & 0x0F)

dig_H6 = getChar(cal3, 6)

# Wait in ms (Datasheet Appendix B: Measurement time and current calculation)
wait_time = 1.25 + (2.3 * OVERSAMPLE_TEMP) + ((2.3 * OVERSAMPLE_PRES) + 0.575) +
((2.3 * OVERSAMPLE_HUM)+0.575)
time.sleep(wait_time/1000)  # Wait the required time

# Read temperature/pressure/humidity
data = bus.read_i2c_block_data(addr, REG_DATA, 8)
pres_raw = (data[0] << 12) | (data[1] << 4) | (data[2] >> 4)
temp_raw = (data[3] << 12) | (data[4] << 4) | (data[5] >> 4)
hum_raw = (data[6] << 8) | data[7]

#Refine temperature
var1 = ((((temp_raw>>3)-(dig_T1<<1)))*(dig_T2)) >> 11
var2 = (((((temp_raw>>4) - (dig_T1)) * ((temp_raw>>4) - (dig_T1))) >> 12) * (dig_T3)) >> 14
t_fine = var1+var2
temperature = float(((t_fine * 5) + 128) >> 8);

# Refine pressure and adjust for temperature
var1 = t_fine / 2.0 - 64000.0
var2 = var1 * var1 * dig_P6 / 32768.0
var2 = var2 + var1 * dig_P5 * 2.0
var2 = var2 / 4.0 + dig_P4 * 65536.0
var1 = (dig_P3 * var1 * var1 / 524288.0 + dig_P2 * var1) / 524288.0
var1 = (1.0 + var1 / 32768.0) * dig_P1
if var1 == 0:
  pressure=0
else:
  pressure = 1048576.0 - pres_raw
  pressure = ((pressure - var2 / 4096.0) * 6250.0) / var1
  var1 = dig_P9 * pressure * pressure / 2147483648.0
  var2 = pressure * dig_P8 / 32768.0
  pressure = pressure + (var1 + var2 + dig_P7) / 16.0

# Refine humidity
humidity = t_fine - 76800.0
```

```
  humidity = (hum_raw - (dig_H4 * 64.0 + dig_H5 / 16384.0 * humidity)) * (dig_H2 / 65536.0 *
(1.0 + dig_H6 / 67108864.0 * humidity * (1.0 + dig_H3 / 67108864.0 * humidity)))
  humidity = humidity * (1.0 - dig_H1 * humidity / 524288.0)
  if humidity > 100:
    humidity = 100
  elif humidity < 0:
    humidity = 0

  return temperature/100.0,pressure/100.0,humidity

def main():

  (chip_id, chip_version) = readBME280ID()
  print "Chip ID     :", chip_id
  print "Version     :", chip_version

  temperature,pressure,humidity = readBME280All()

  print "Temperature : ", temperature, "C"
  print "Pressure : ", pressure, "hPa"
  print "Humidity : ", humidity, "%"

while True:
  (chip_id, chip_version) = readBME280ID()
  print "Chip ID     :", chip_id
  print "Version     :", chip_version
  temperature,pressure,humidity = readBME280All()
  print "Temperature : ", temperature, "C"
  print "Pressure : ", pressure, "hPa"
  print "Humidity : ", humidity, "%"
  time.sleep(2)

if __name__=="__main__":
  main()
```

BMI160:

ETS IoT KIT has ACCELEROMETER/GYROSCOPE which has been connected to raspberry pi through I2C. It will show the value of ACCELEROMETER and GYROSCOPE.This sensor is connected with ETS IoT KIT Raspberry pi thorough I2C address 0x68

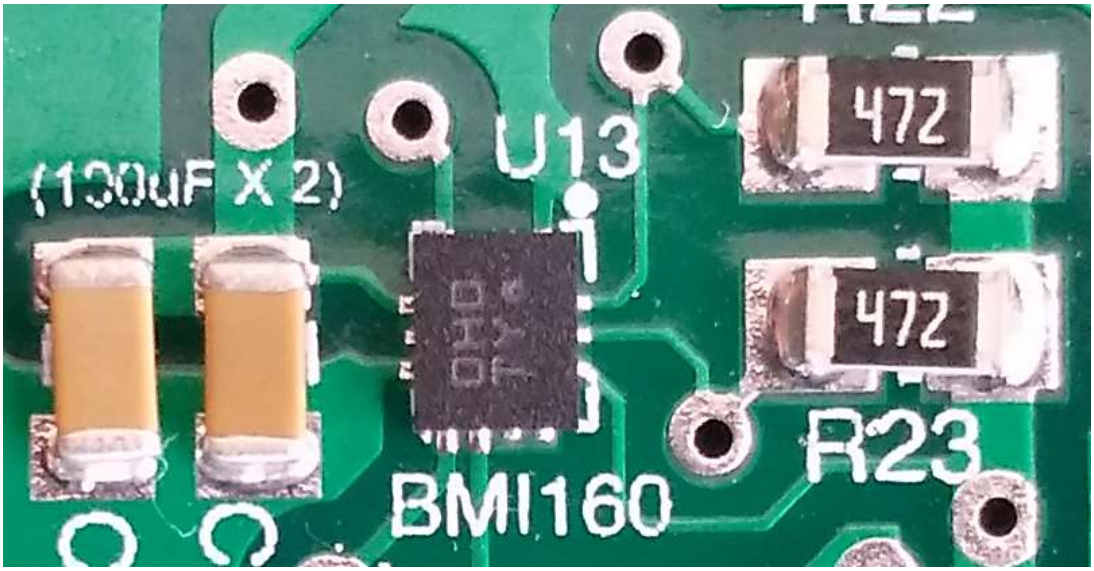


Figure: BMI160 image in ETS-IoT Trainer Kit

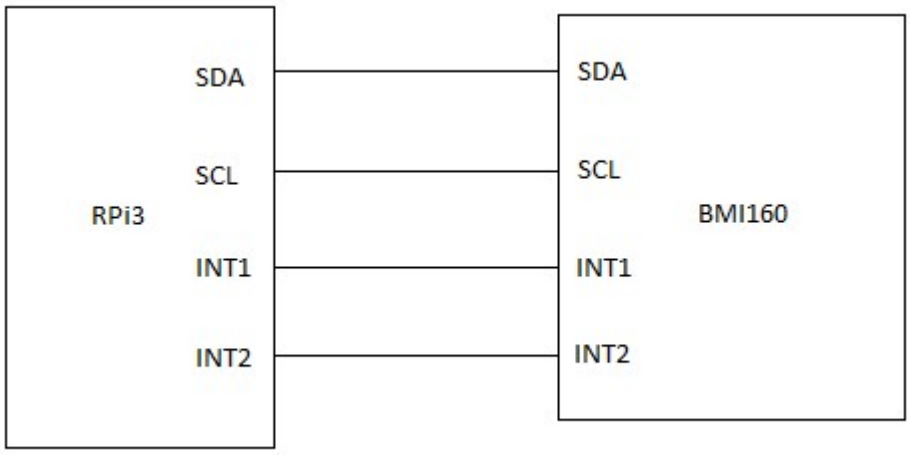


Figure: Connection diagram of BMI160

Software Example:

```
# Application to read and display the value of ACCELEROMETER and GYROSCOPE.

import smbus
import time
import os

DEVICE              = 0x68 # Default device I2C address
ACCEL_MODE_NORMAL       = 0x11
GYRO_MODE_NORMAL        = 0x15
ACCEL_RANGE_2G          = 0x03
ACC_SCALE               = 16384.0
GYRO_SCALE              = 16.4

# Register definitions
DEVICE_ID       = 0x00
COMMAND_REG     = 0X7E
ACCEL_RANGE     = 0x41
ACCEL_DATAX_LSB  = 0x12
GYRO_DATAX_LSB   = 0x0C

bus = smbus.SMBus(1)

def CheckDeviceID():
  # Read and print device ID
  #print "BMI160 device ID :  ",bus.read_i2c_block_data(DEVICE, DEVICE_ID, 1)
  chip_id = bus.read_i2c_block_data(DEVICE, DEVICE_ID, 1)
  print "Chip ID     :", chip_id
  return chip_id

def InitSensor():
  # Put sensor into normal mode as it is by default in standby mode
  # also some time delay is required after each write
  bus.write_byte_data(DEVICE,COMMAND_REG,ACCEL_MODE_NORMAL)
  time.sleep(1)

  bus.write_byte_data(DEVICE,COMMAND_REG,GYRO_MODE_NORMAL)
  time.sleep(1)

def ReadSensor():
```

```
    # Read six bytes of accel data from register 0X12 to 0x17
    bytes = bus.read_i2c_block_data(DEVICE, ACCEL_DATAX_LSB, 6)

    x = bytes[0] | (bytes[1] << 8)
    if(x & (1 << 16 - 1)):
      x = x - (1<<16)

    y = bytes[2] | (bytes[3] << 8)
    if(y & (1 << 16 - 1)):
      y = y - (1<<16)

    z = bytes[4] | (bytes[5] << 8)
    if(z & (1 << 16 - 1)):
      z = z - (1<<16)

    Ax = round((x / ACC_SCALE),2)
    Ay = round((y / ACC_SCALE),2)
    Az = round((z / ACC_SCALE),2)

    return Ax,Ay,Az

def Read1Sensor():
    # Read six bytes of gyro data from register 0X0C
    bytes = bus.read_i2c_block_data(DEVICE, GYRO_DATAX_LSB, 6)

    x = bytes[0] | (bytes[1] << 8)
    if(x & (1 << 16 - 1)):
      x = x - (1<<16)

    y = bytes[2] | (bytes[3] << 8)
    if(y & (1 << 16 - 1)):
      y = y - (1<<16)

    z = bytes[4] | (bytes[5] << 8)
    if(z & (1 << 16 - 1)):
      z = z - (1<<16)

    Gx = round((x / GYRO_SCALE),2)
    Gy = round((y / GYRO_SCALE),2)
    Gz = round((z / GYRO_SCALE),2)

    return Gx,Gy,Gz

def main():
```

```
    chip_id = CheckDeviceID()
    InitSensor()
    Ax,Ay,Az = ReadSensor()
    Gx,Gy,Gz = Read1Sensor()


if __name__=="__main__":
    main()
while True:
  # read sensor value
  Ax,Ay,Az = ReadSensor()
  print "ACC X",Ax
  print "ACC Y",Ay
  print "ACC Z",Az
  print"\n"
  Gx,Gy,Gz = Read1Sensor()
  print "GCC X",Gx
  print "GCC Y",Gy
  print "GCC Z",Gz
  print"\n"
  time.sleep(1)
```

RELAY CONTROL:

ETS IoT KIT has 3 RELAY which has been connected to raspberry pi.

Warning:

User application may use ETS IoT KIT relays directly 230 volts, care must be taken by the user so that the hardware is not damaged. Any misuse will damage complete kit and it is not covered under warranty

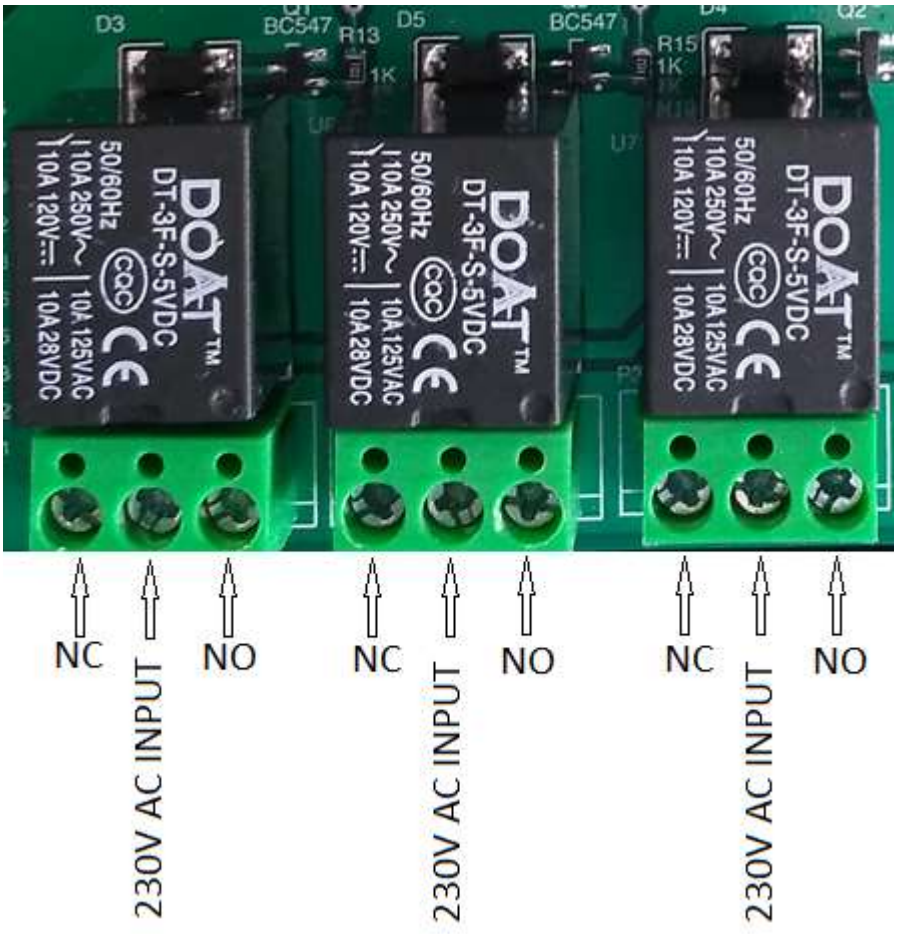


Figure: RELAY Image in ETS-IoT Trainer Kit

Note:
NC – Normally Closed, NO – Normally Open
User has to connect 230V supply as indicated in below connection representation.

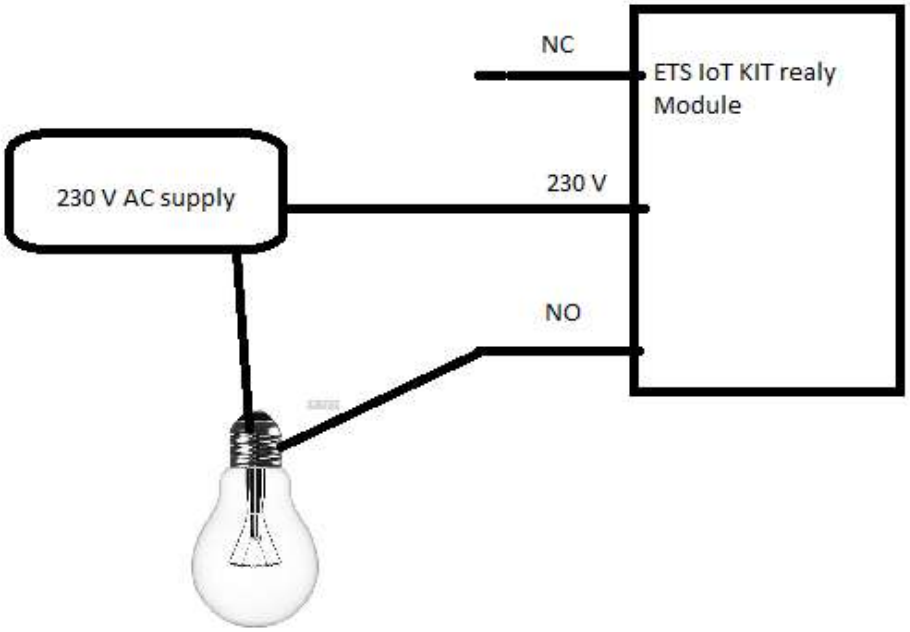


Figure: Connection diagram of RELAY with Bulb

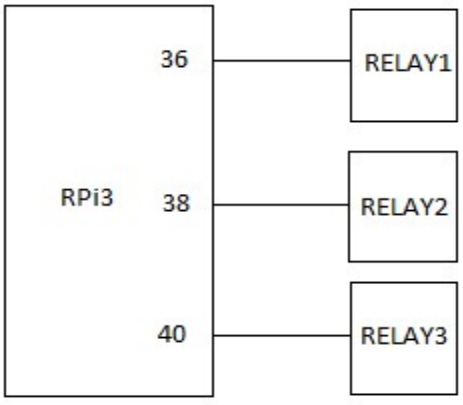


Figure: Connection diagram of RELAY

Software Example:

```
#Application to ON and OFF the RELAY which has connected to pin no.36. User can control
#remaining RELAY which has connected with pin no.38 and 40.

#Raspberry pi GPIO configuration
import RPi.GPIO as GPIO
import time


# Raspberry pi input/output configuration
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(36, GPIO.OUT)

while True:

    #Relay1 ON
    GPIO.output(36, 1)
    time.sleep(2)

    #Relay1 OFF
    GPIO.output(36, 0)
    time.sleep(2)
```

OLED DISPLAY:

ETS IoT KIT has an OLED DISPLAY which has been connected to raspberry pi through I2C address 0x3C

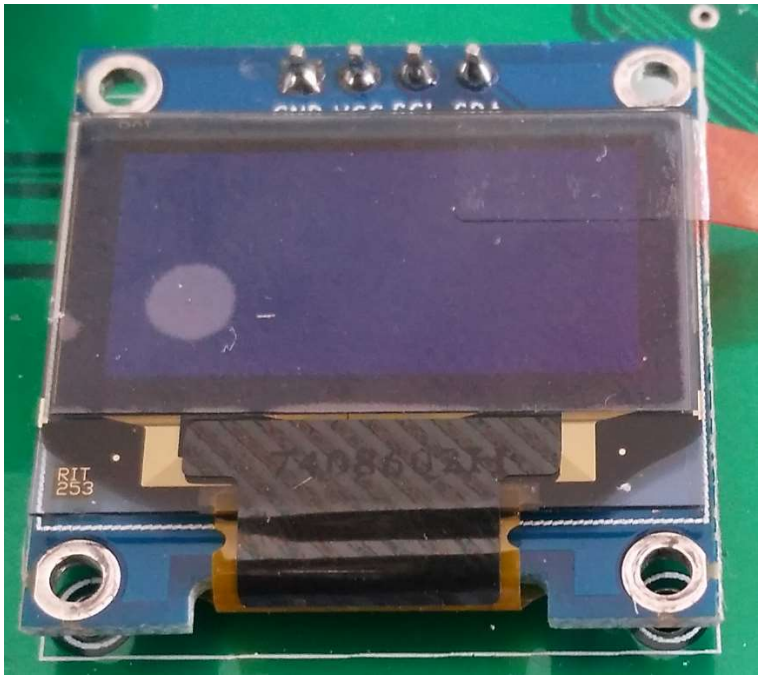


Figure: OLED Display Image in ETS-IoT Trainer Kit

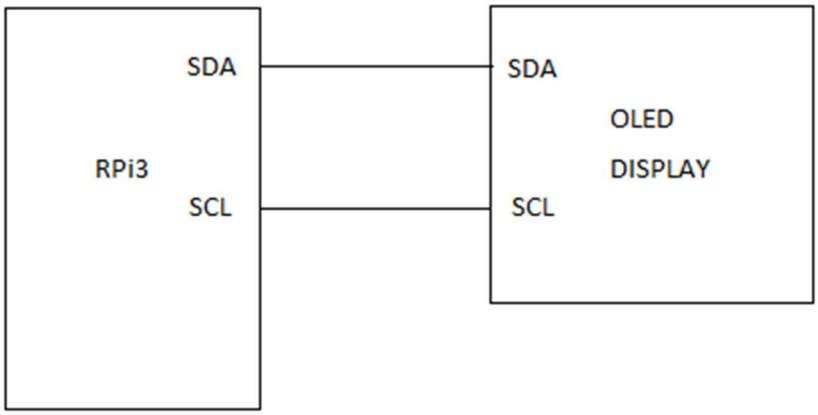


Figure: Connection diagram of OLED Display

Software Example:

```
# Application to display two lines of text. User can modify and print something.

import time
import Adafruit_SSD1306

from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont

# Raspberry Pi pin configuration:
RST = 24
disp = Adafruit_SSD1306.SSD1306_128_64(rst=RST, i2c_address=0x3C)

# Initialize library.
disp.begin()

# Clear display.
disp.clear()
disp.display()

#Make sure to create image with mode '1' for 1-bit color.
width = disp.width
height = disp.height
image = Image.new('1', (width, height))

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)


# Draw a black filled box to clear the image.
draw.rectangle((0,0,width,height), outline=0, fill=0)

# Draw some shapes.
### First define some constants to allow easy resizing of shapes.
padding = 2
top = padding


# Move left to right keeping track of the current x position for drawing shapes.
x = padding
```

```
# Load default font.
font = ImageFont.load_default()

#Write two lines of text.
draw.text((x, top),    'HELLO...',  font=font, fill=255)
draw.text((x, top+20), 'WORLD!', font=font, fill=255)
#Display image.
disp.image(image)
disp.display()
time.sleep(5)
```

ONBOARD INPUT / OUTPUT PINS:

ETS IoT KIT has 20 DIGITAL I/O Pins which has been connected to raspberry pi through MCP23017-U11, It has been connected through I2C address 0x21. User can connect any digital sensor for their applications.

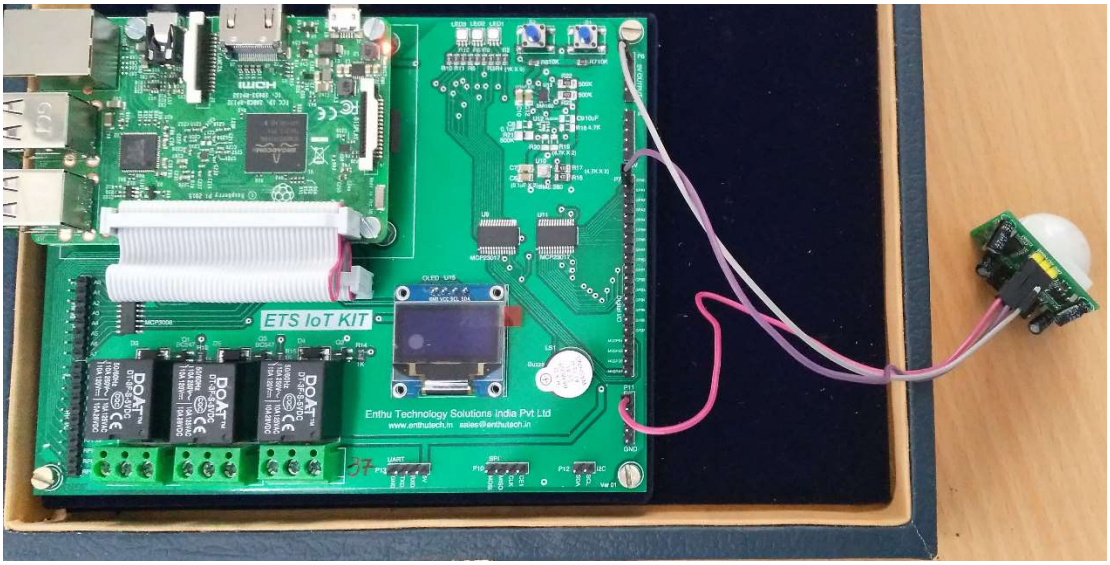


Figure: EXTERNAL DIGITAL Sensor in ETS-IoT Trainer Kit

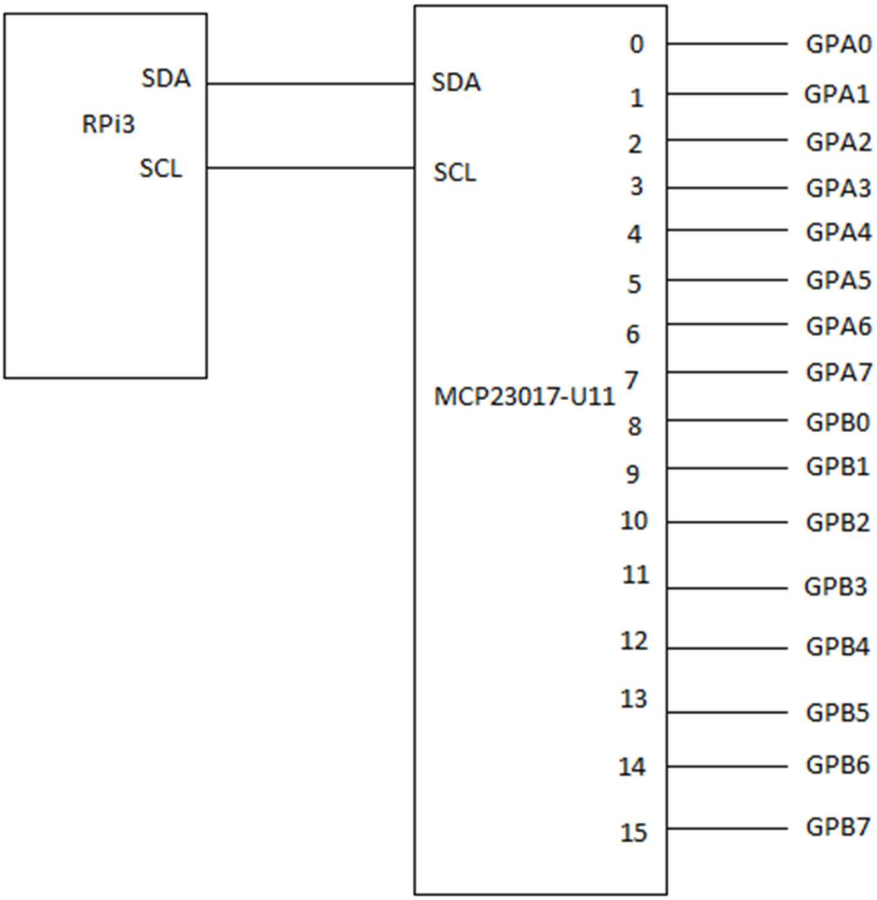


Figure: Connection diagram of External I/O through MCP23017-U11

Software Example:

```
# Application to connect a PIR(Digital) sensor and print the status of Human presence.
# connect LED with GPA0 and mention I2C address=21

import sys
sys.path.append('/home/pi/Adafruit-Raspberry-Pi-Python-Code-legacy/Adafruit_MCP230xx')

from Adafruit_MCP230XX import Adafruit_MCP230XX
import time

mcp = Adafruit_MCP230XX(busnum = 1, address = 0x21, num_gpios = 16)
```

```
# Set pins 0, 1 and 2 to output (you can set pins 0..15 this way)
mcp.config(1, mcp.INPUT)
mcp.pullup(1, 1)

while (True):
   i = mcp.input(1)
   time.sleep(1)

   if i == 1:
      print "person Available"

   if i == 0:
      print "person NOT Available"
```

ANALOG INTERFACE:

ETS IoT KIT has 8 ANALOG Input Pins which has been connected to raspberry pi through MCP3008 through SPI channel 0. User can connect any analog sensor for their applications.

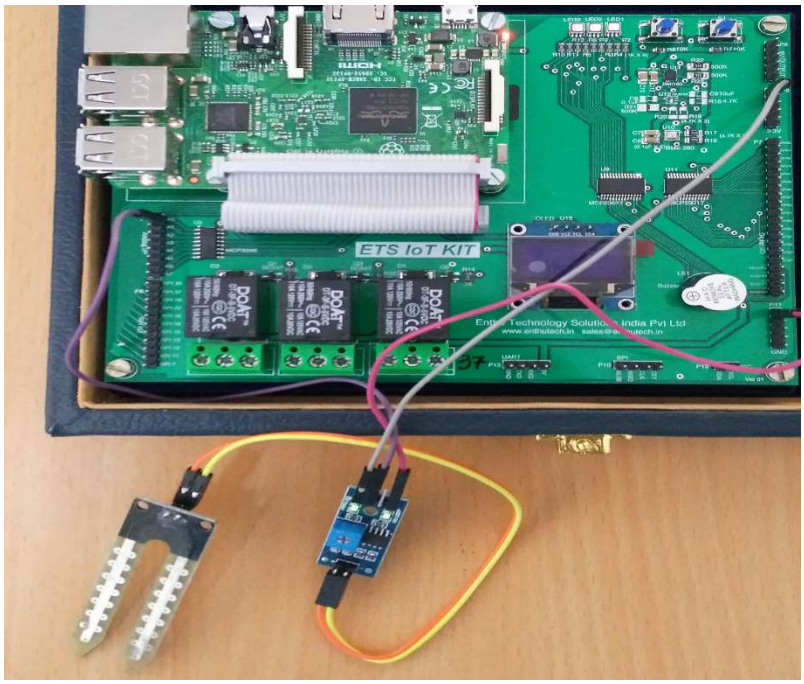


Figure: MOISTURE Sensor in ETS-IoT Trainer Kit

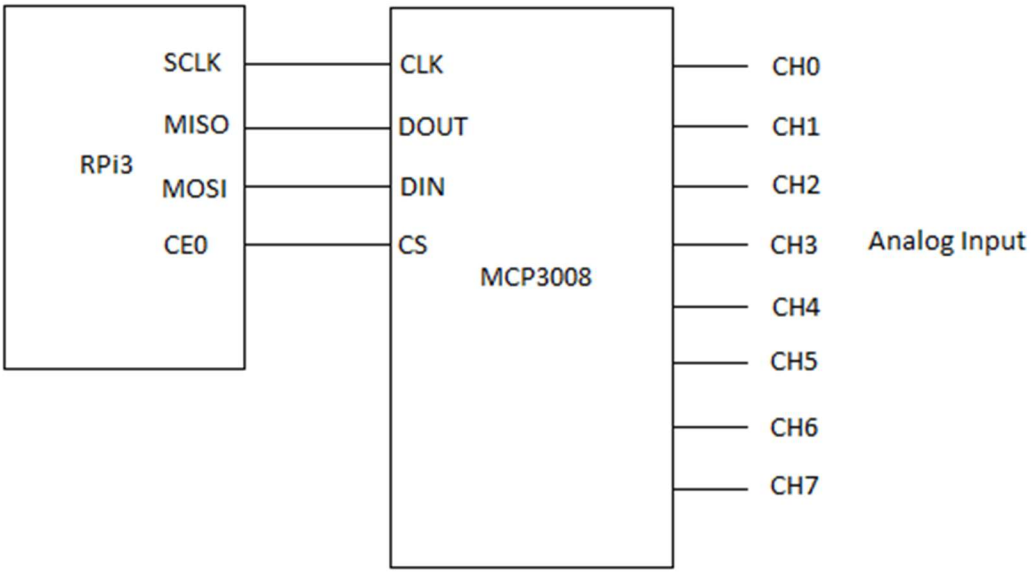


Figure: Connection diagram of Analog Inputs through MCP3008

Software Example:

```
# Application to read and print value of Soil Moisture.

import RPi.GPIO as GPIO
import spidev
import time

spi = spidev.SpiDev()
spi.open(0,0)


def ReadChannel(channel):
  adc = spi.xfer2([1,(8+channel)<<4,0])
  data= ((adc[1]&3) << 8) + adc[2]
  return data

def ConvertVolts(data,places):
  volts = (data * 3.3) / float(1023)
  volts = round(volts,places)
  return volts


light_channel = 0 #CONNECT ANALOG INPUT A0
delay = 3

while True:

  light_level = ReadChannel(light_channel)
  light_volts = ConvertVolts(light_level,2)


  print "_____"
  print ("Light: {} ({}V)".format(light_level,light_volts))
```

DEVICE TO DEVICE COMMUNICATION:

ETS IoT KIT has Wifi , Bluetooth/ Bluetooth Low energy (4.1) capability. User could develop their own applications on any of these radio technologies. In below section describes a Machine to Machine (M2M) example application with Bluetooth.

Step 1:

First you have to pair the two devices.
Open the LX-Terminal in two Raspberry pi and enter the below commands one by one.

bluetoothctl
power on
agent on
default-agent
discoverable on
scan on

after the scan command it will show the nearby Bluetooth devices

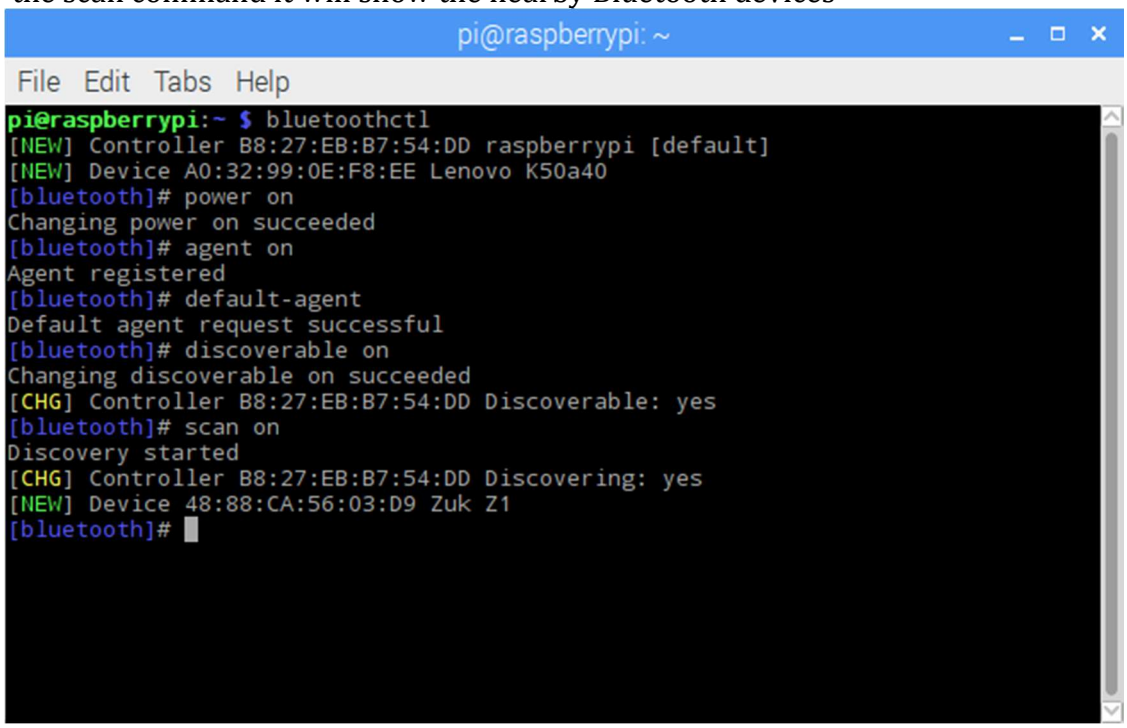


Figure: LxTerminal capture for Bluetooth nearby devices

Step 2:

Then pair the device with your client using the below command.
pair <MAC address of client>

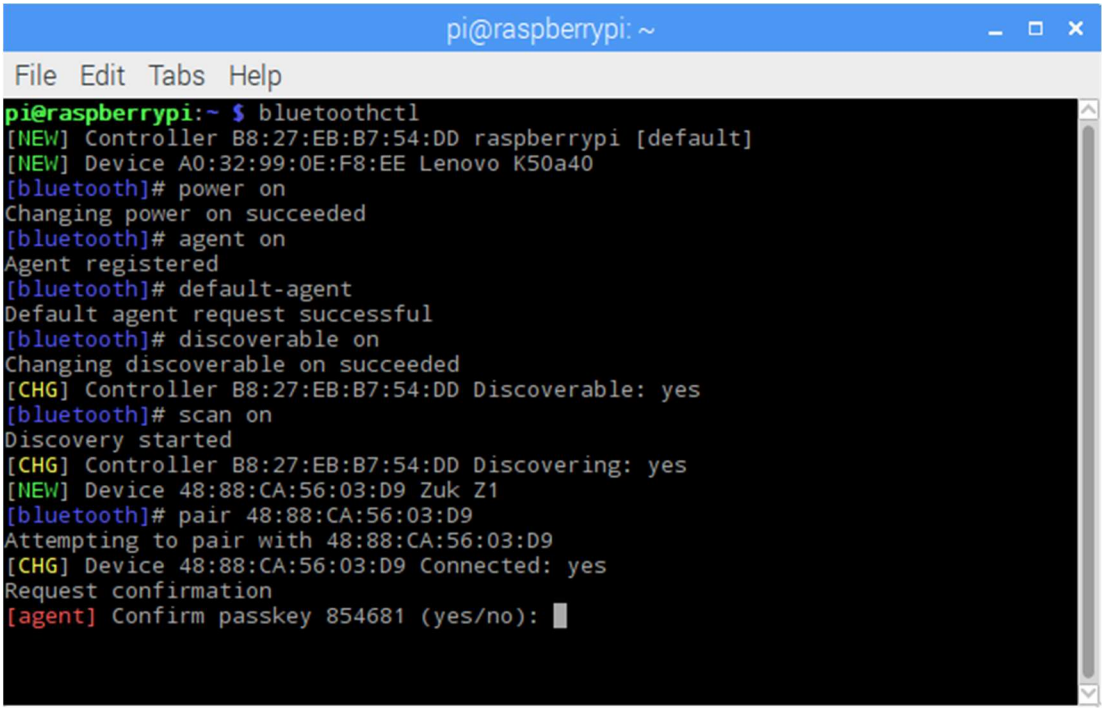


Figure: LxTerminal capture for paring nearby devices

It will ask pairing request and click OK to connect.

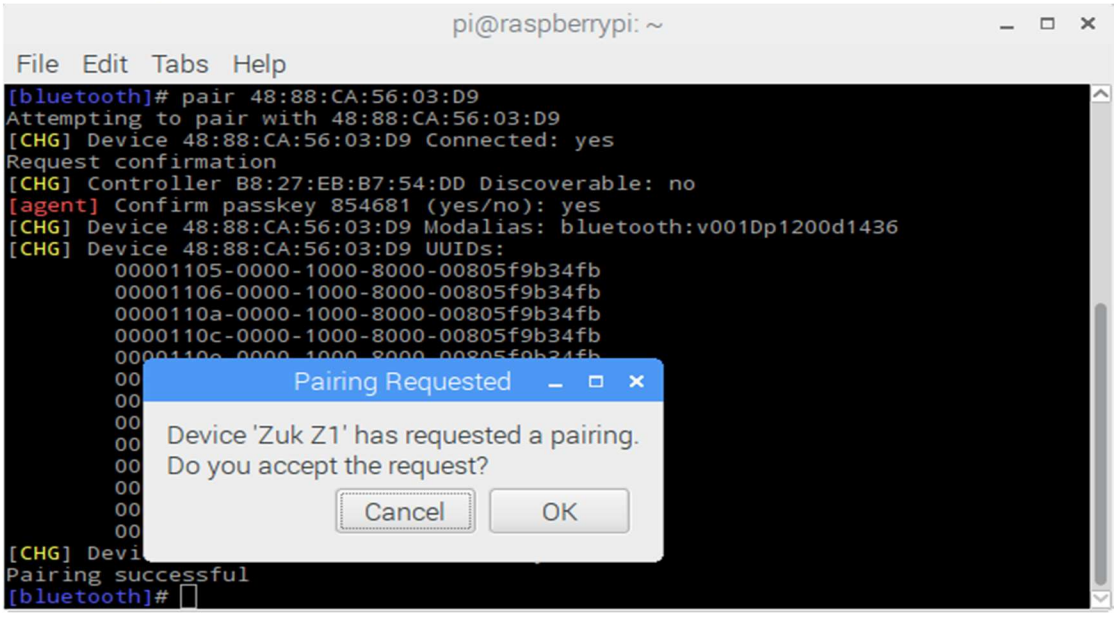


Figure: LxTerminal capture for paring nearby devices

Once devices got connected you can see the below window.

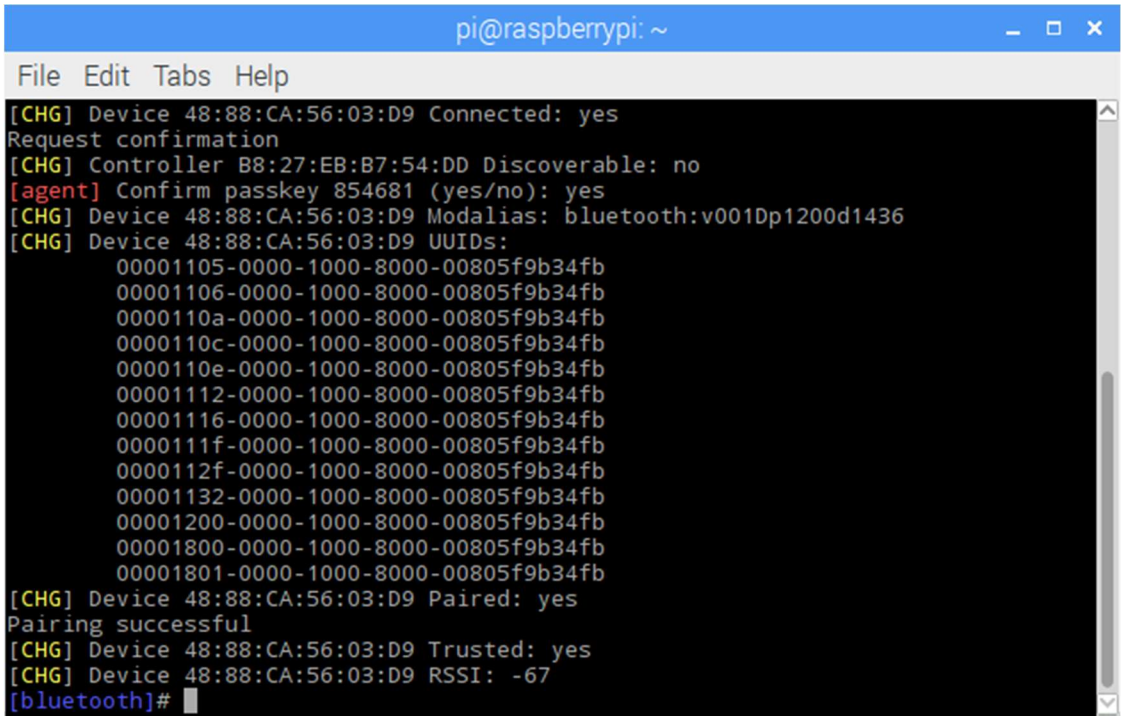


Figure: LxTerminal capture for successful paring

If you want to list out the paired devices enter the below command
Paired-devices

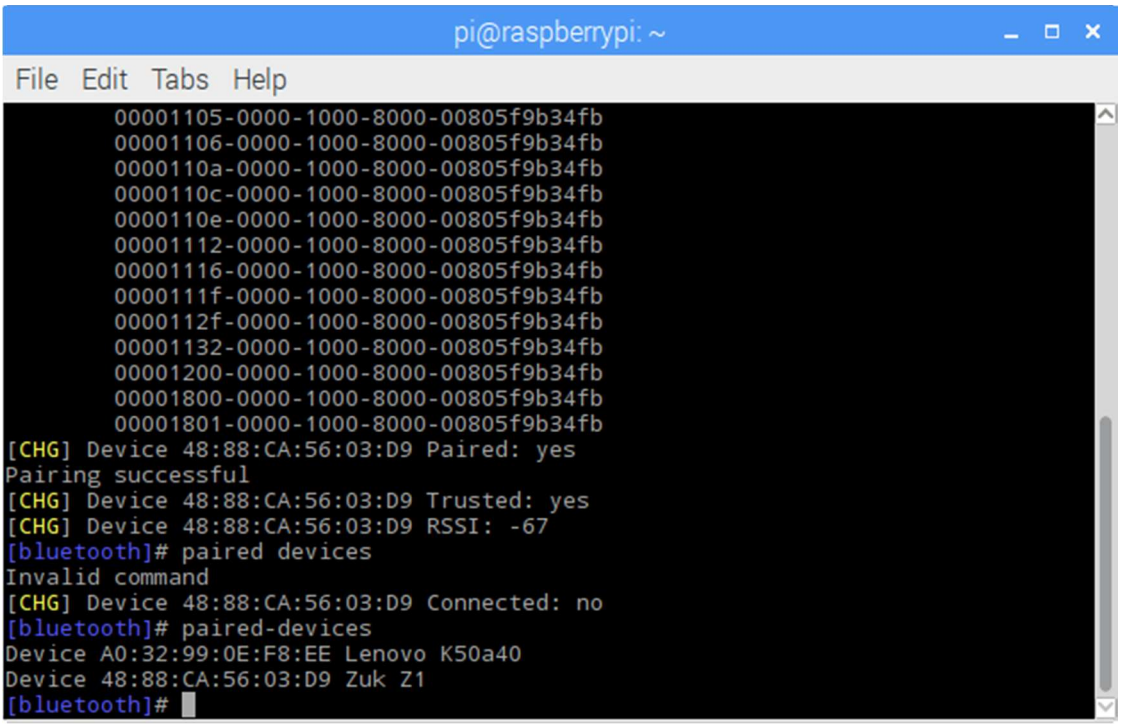


Figure: LxTerminal capture for listing paired devices

First open and run the python script from Server device. Then open and run the python script from client device.

Software Example Bluetooth server:

```
# Program for Bluetooth Server which receives/sends commands from client and turn ON / OFF the LEDs

import bluetooth
import time
import sys
```

```
sys.path.append('/home/pi/Adafruit-Raspberry-Pi-Python-Code-legacy/Adafruit_MCP230xx')

server_sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )

port = 1
server_sock.bind(("", port))
server_sock.listen(1)

client_sock,address = server_sock.accept()
print "Accepted connection from", address

from Adafruit_MCP230XX import Adafruit_MCP230XX

mcp = Adafruit_MCP230XX(busnum = 1, address = 0x20, num_gpios = 16)

mcp.config(0, mcp.OUTPUT)
mcp.config(9, mcp.INPUT)
mcp.pullup(9, 1)

while True:
   data = client_sock.recv(1024)
   print "received [%s]" % data
   if data == 'ON':
      mcp.output(0, 1)
      print "LED ON"

   if data == 'OFF':
      mcp.output(0, 0)
      print "LED OFF"
   time.sleep(0.5)

   input = (mcp.input(9))
   if input == 0:
      text = "ON"
      client_sock.send(text)
   if input == 512:
      text = "OFF"
      client_sock.send(text)
   time.sleep(0.5)

client_sock.close()
server_sock.close()
```

Software Example Bluetooth client:

```
# Program for Bluetooth client which receives/sends commands from server and turn ON / OFF the LEDs

import bluetooth
import sys
sys.path.append('/home/pi/Adafruit-Raspberry-Pi-Python-Code-legacy/Adafruit_MCP230xx')

bd_addr = "B8:27:EB:87:EB:3E" #server MAC address
port = 1
sock = bluetooth.BluetoothSocket( bluetooth.RFCOMM )
sock.connect((bd_addr,port))

from Adafruit_MCP230XX import Adafruit_MCP230XX
import time

mcp = Adafruit_MCP230XX(busnum = 1, address = 0x20, num_gpios = 16)

mcp.config(0, mcp.OUTPUT)
mcp.config(9, mcp.INPUT)
mcp.pullup(9, 1)

while True:
   input = (mcp.input(9))
   if input == 0:
      text = "ON"
      sock.send(text)

   if input == 512:
      text = "OFF"
      sock.send(text)
   time.sleep(0.2)

   data = sock.recv(1024)
   print "received [%s]" % data
```

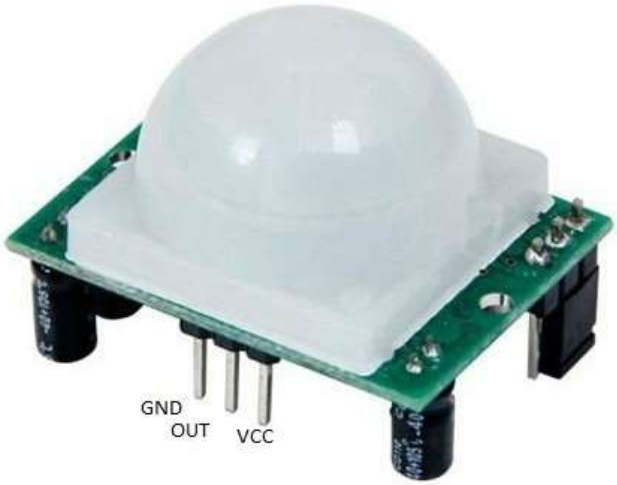
```
    if data == 'ON':
        mcp.output(0, 1)
        print "LED ON"

    if data == 'OFF':
        mcp.output(0, 0)
        print "LED OFF"
    time.sleep(0.2)

sock.close()
```

EXTERNAL SENSOR INTERFACING

ETS IoT KIT offers 31 GPIOs to connect and play with external sensors, below section describes about various external sensor connections with software examples.

PASSIVE INFRA RED:

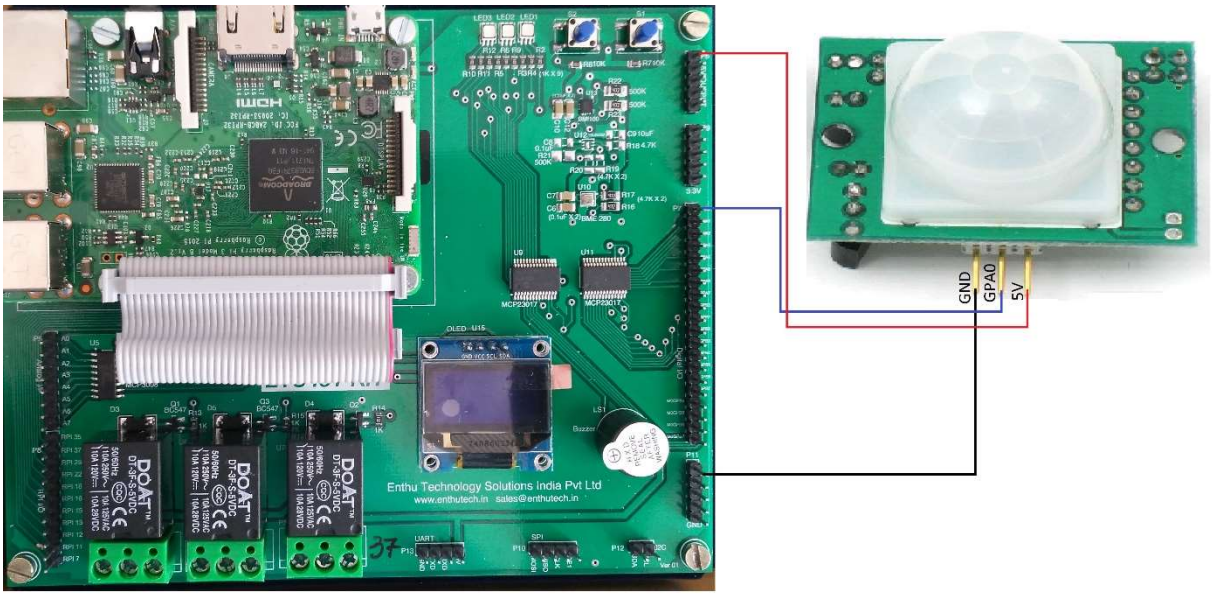
A passive infrared sensor (PIR sensor) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors.



Applications of PIR Sensors

- All outdoor Lights
- Lift Lobby
- Multi Apartment Complexes
- Common staircases
- For Basement or Covered Parking Area
- Shopping Malls
- For garden lights

Connection diagram of PIR:



Pin mapping:

ETS-IoT Kit	SENSOR
5V	VCC
GPA0	OUTPUT
GND	GND

Software Example:

```
# Application to connect a PIR(Digital) sensor and print the status of Human presence.


import sys
sys.path.append('/home/pi/Adafruit-Raspberry-Pi-Python-Code-
legacy/Adafruit_MCP230xx')

from Adafruit_MCP230XX import Adafruit_MCP230XX
import time

mcp = Adafruit_MCP230XX(busnum = 1, address = 0x21, num_gpios = 16)

# Set pins 0, 1 and 2 to output (you can set pins 0..15 this way)
mcp.config(0, mcp.INPUT)
mcp.pullup(0, 1)

while (True):
   i = mcp.input(0)
   time.sleep(1)

   if i == 1:
      print "person detect"

   if i == 0:
      print "person not detect"
```

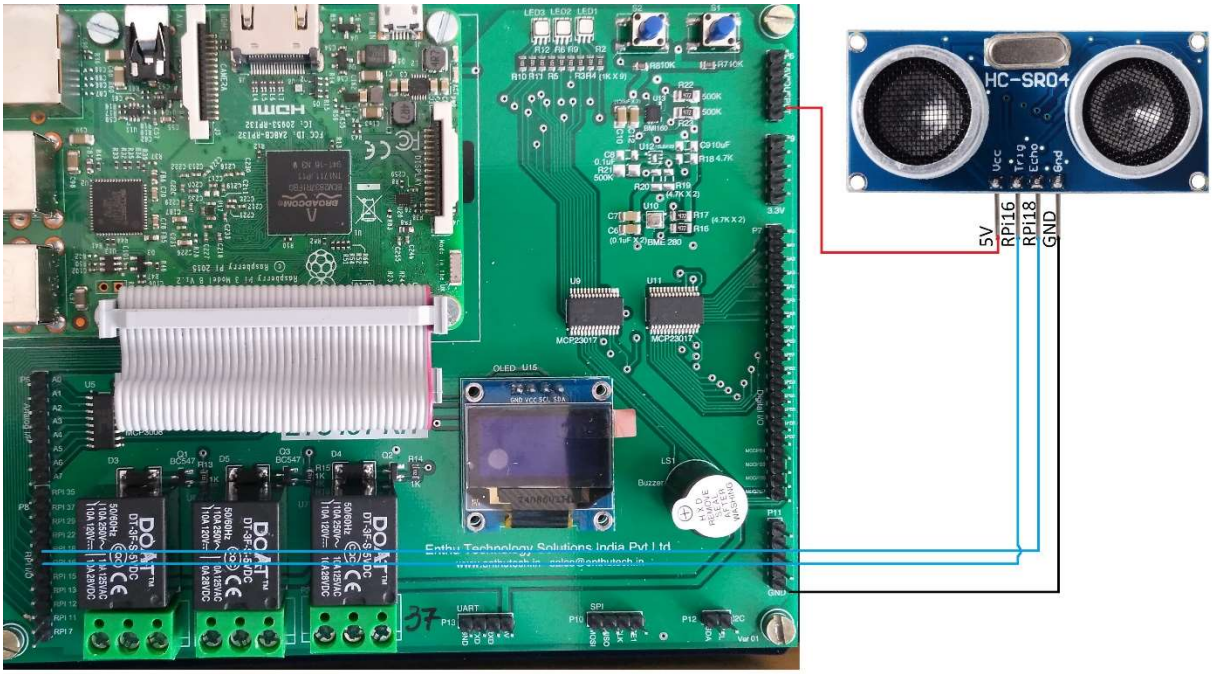
ULTRASONIC SENSOR:

Ultrasonic sensors "are based on the measurement of the properties of acoustic waves with frequencies above the human audible range," often at roughly 40 kHz 1). They typically operate by generating a high-frequency pulse of sound, and then receiving and evaluating the properties of the echo pulse.

Applications of ULTROSONIC Sensors

- People detection for counting
- Vehicle detection for car wash and automotive assembly
- Robotic sensing
- Stacking height control

Connection diagram of ULTRASONIC Sensor:



Pin mapping:

ETS-IoT Kit	SENSOR
5V	VCC
RPi16	TRIG
RPi18	ECHO
GND	GND

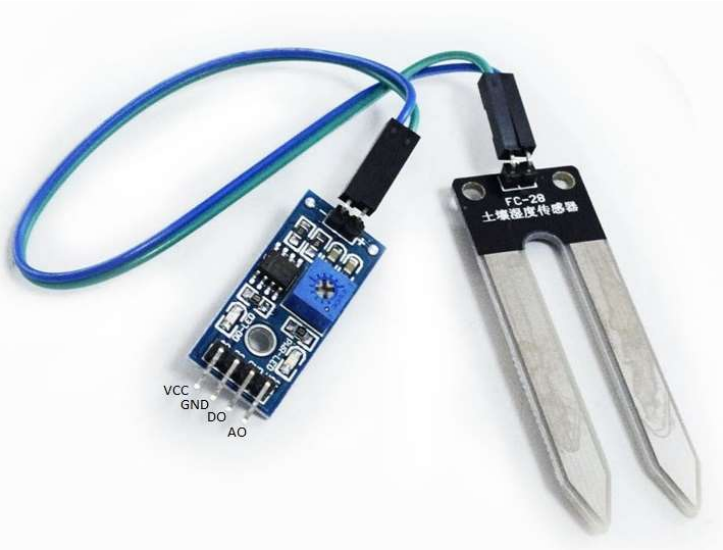
Software Example:

```
# Application to connect a ULTRASONIC(Digital) sensor and print the distance of object.

import sys

import time
import RPi.GPIO as GPIO

# Use BCM GPIO references
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)

# Define GPIO to use on Pi
GPIO_TRIGGER = 16 ##connect with RPI16
GPIO_ECHO    = 18 ##connect with RPI18

# Set pins as output and input
GPIO.setup(GPIO_TRIGGER,GPIO.OUT)  # Trigger
GPIO.setup(GPIO_ECHO,GPIO.IN)      # Echo

# Set trigger to False (Low)
GPIO.output(GPIO_TRIGGER, False)

# Allow module to settle
time.sleep(0.5)
while True:
  # Send 10us pulse to trigger
  GPIO.output(GPIO_TRIGGER, True)
  time.sleep(0.00001)
  GPIO.output(GPIO_TRIGGER, False)
  #start = time.time()

  while GPIO.input(GPIO_ECHO)==0:
    start = time.time()
```

```
while GPIO.input(GPIO_ECHO)==1:
  stop = time.time()

# Calculate pulse length
elapsed = stop-start

# Distance pulse travelled in that time is time
# multiplied by the speed of sound (cm/s)
distance = elapsed * 34300

# That was the distance there and back so halve the value
distance = distance / 2
print "Distance : %.1f" % distance
# Reset GPIO settings
#GPIO.cleanup()
```

SOIL MOISTURE SENSOR:

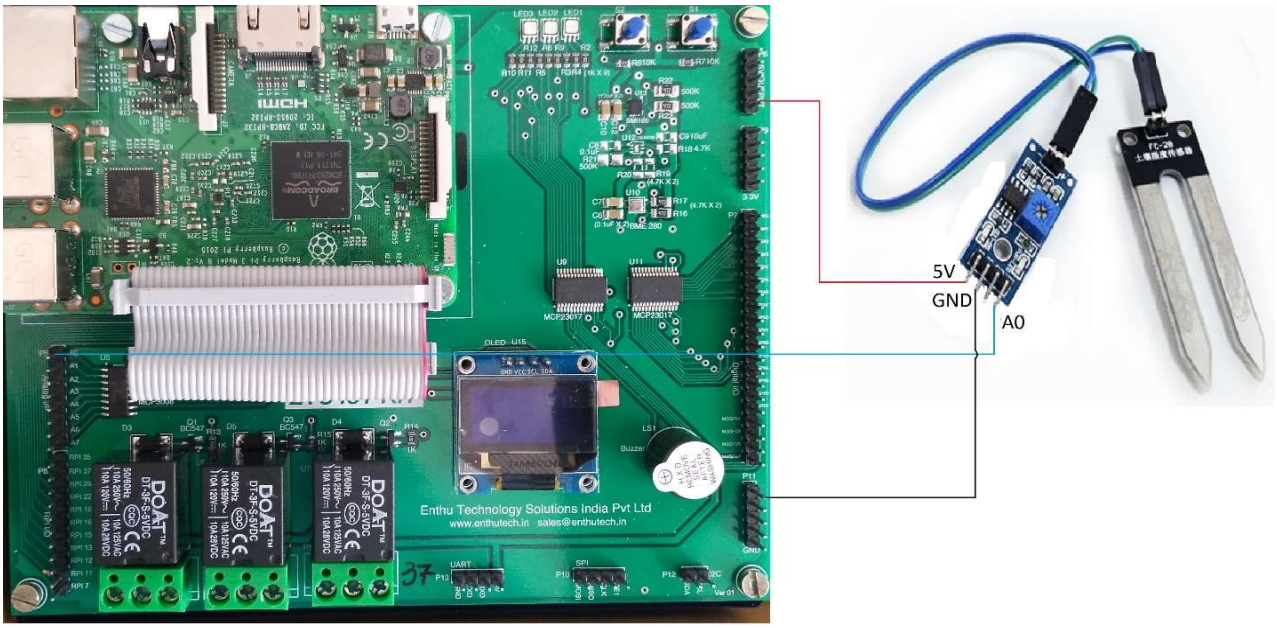
Soil Moisture Sensor is a simple breakout for measuring the moisture in soil and similar materials. The soil moisture sensor is pretty straight forward to use. The two large exposed pads function as probes for the sensor, together acting as a variable resistor. The more water that is in the soil means the better the conductivity between the pads will be and will result in a lower resistance, and a higher SIG out.



Applications of SOIL MOISTURE Sensors

- Agriculture

Connection diagram of SOIL MOISTURE Sensor:



Pin mapping:

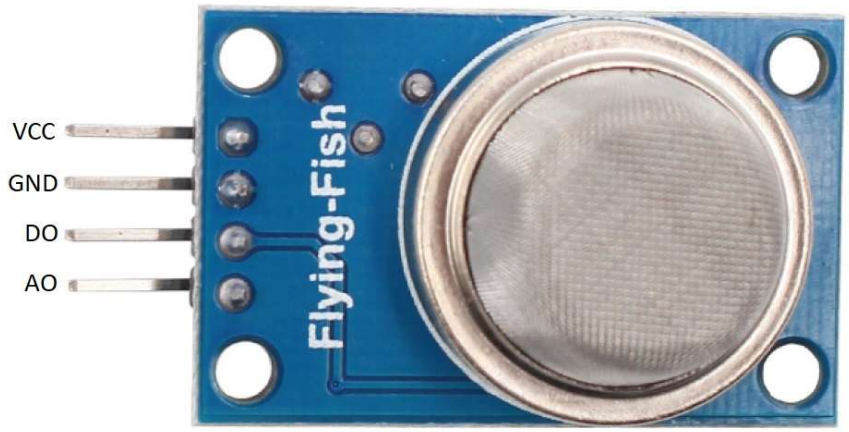
ETS-IoT Kit	SENSOR
5V	VCC
A0	AO
GND	GND

Software Example:

```
# Application to connect a SOIL MOISTURE(Analog) sensor and print the value of soilmoisture.

import spidev
import time

spi = spidev.SpiDev()
spi.open(0,0)


def ReadChannel(channel):
  adc = spi.xfer2([1,(8+channel)<<4,0]) #as per datasheet
  data= ((adc[1]&3) << 8) + adc[2]
  return data

def ConvertVolts(data,places):
  volts = (data * 3.3) / float(1023)
  volts = round(volts,places)
  return volts


moisture_channel = 0 ##connect with A0
delay = 3

while True:


  moisture_level = ReadChannel(moisture_channel)
  moisture_volts = ConvertVolts(moisture_level,2)

  print "_____"
  print ("Moisture: {} ({}V)".format(moisture_level,moisture_volts))

  time.sleep(delay)
```

GAS SENSOR:

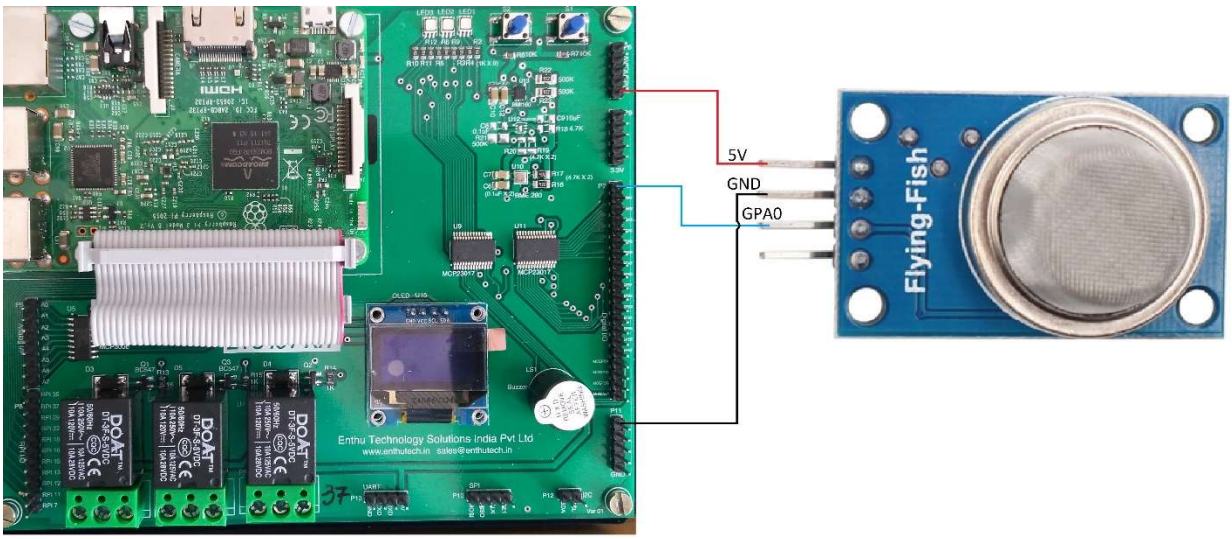
Gas Sensor (MQ2) module is useful for gas leakage detecting (in home and industry). It can detect LPG, i-butane, methane, alcohol, Hydrogen, smoke and so on. Based on its fast response time. Measurements can be taken as soon as possible. Also the sensitivity can be adjusted by the potentiometer.



Applications of GAS Sensors:

- Gas leakage detection

Connection diagram of GAS Sensor:



Pin mapping:

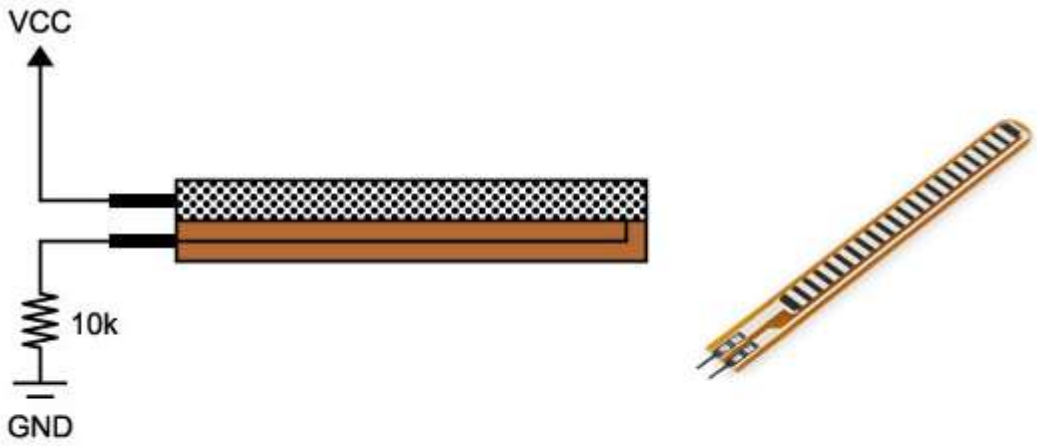
ETS-IoT Kit	SENSOR
5V	VCC
GPA0	D0
GND	GND

Software Example:

```
# Application to connect a GAS(Digital) sensor and print the status of GAS Leakage.

import sys
sys.path.append('/home/pi/Adafruit-Raspberry-Pi-Python-Code-
legacy/Adafruit_MCP230xx')

from Adafruit_MCP230XX import Adafruit_MCP230XX
import time

mcp = Adafruit_MCP230XX(busnum = 1, address = 0x21, num_gpios = 16)


# Set pins 0, 1 and 2 to output (you can set pins 0..15 this way)
mcp.config(0, mcp.INPUT)
mcp.pullup(0, 1)


while (True):
   i = mcp.input(0)
   time.sleep(1)

   if i == 1:
      print "GAS detect"

   if i == 0:
      print "GAS not detect"
```

FLEX SENSOR:

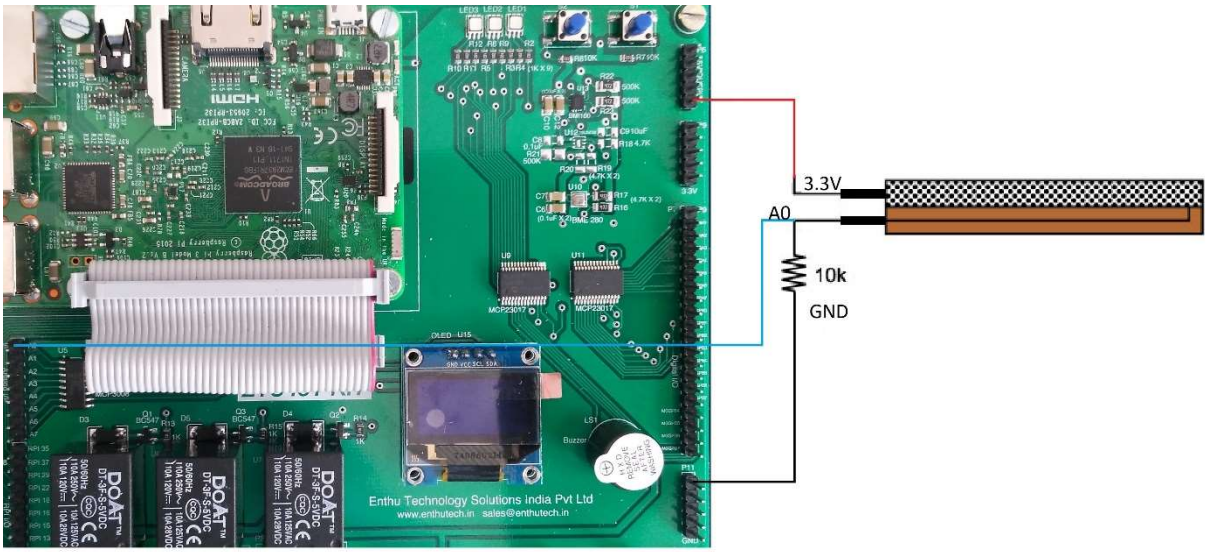
Flex sensor or bend sensor is a sensor that measures the amount of deflection or bending. Usually, the sensor is stuck to the surface, and resistance of sensor element is varied by bending the surface. Since the resistance is directly proportional to the amount of bend it is used as goniometer, and often called flexible potentiometer.



Applications of FLEX Sensors:

- Robotics
- Bio-metrics
- Virtual Reality Gaming Gloves

Connection diagram of FLEX Sensor:



Pin mapping:

ETS-IoT Kit	SENSOR
5V	VCC
A0	AO
GND	GND

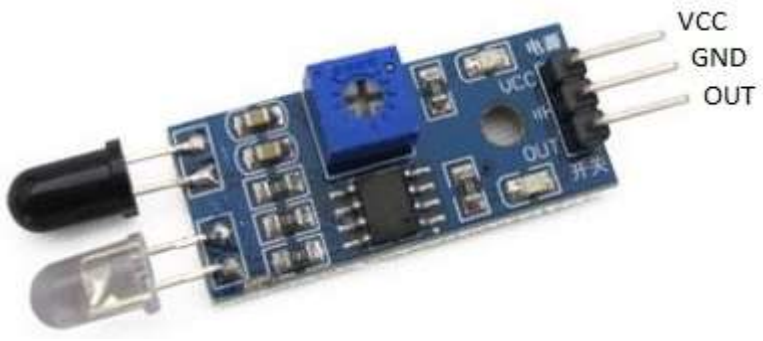
Software Example:

```
# Application to connect a FLEX(Analog) sensor and print the amount of deflection.

import spidev
import time

spi = spidev.SpiDev()
spi.open(0,0)


def ReadChannel(channel):
  adc = spi.xfer2([1,(8+channel)<<4,0]) #as per datasheet
  data= ((adc[1]&3) << 8) + adc[2]
```

```
    return data

def ConvertVolts(data,places):
    volts = (data * 3.3) / float(1023)
    volts = round(volts,places)
    return volts


flex_channel = 0 ##connect with A0
delay = 3

while True:


    flex_level = ReadChannel(flex_channel)
    flex_volts = ConvertVolts(flex_level,2)

    print "_____"
    print ("Moisture: {} ({}V)".format(flex_level, flex_volts))

    time.sleep(delay)
```

IR SENSOR:

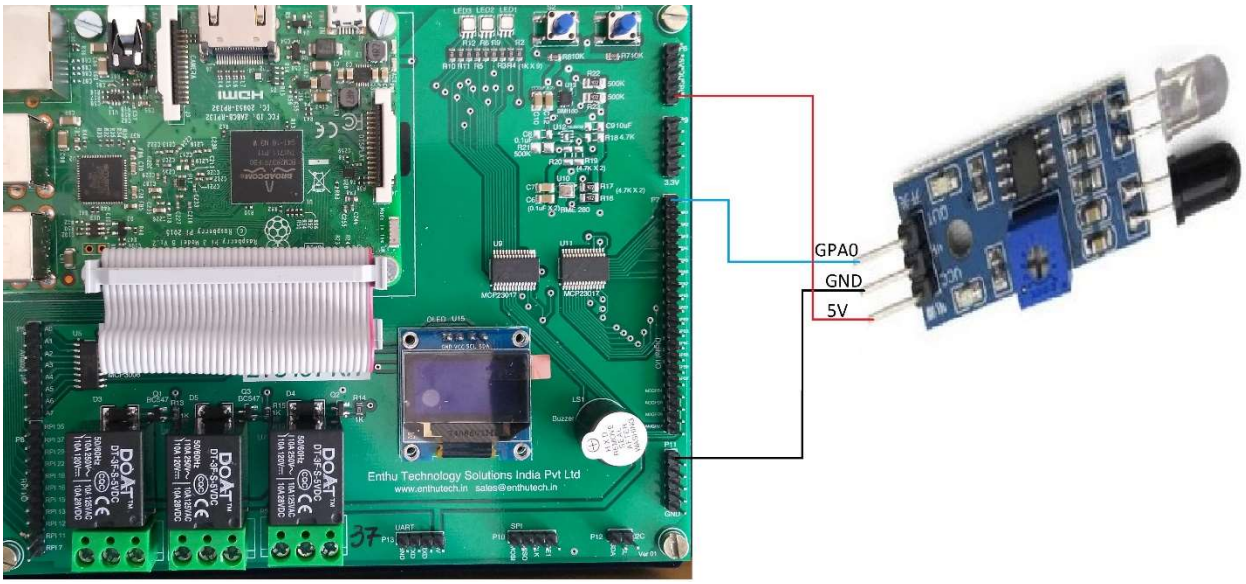
Infrared Obstacle Sensor Module has built-in IR transmitter and IR receiver that sends out IR energy and looks for reflected IR energy to detect presence of any obstacle in front of the sensor module. The module has on board potentiometer that lets user adjust detection range. The sensor has very good and stable response even in ambient light or in complete darkness.



Applications of IR Sensors:

- Line Follower Robots
- Item Counter
- Proximity Sensors
- Burglar Alarm

Connection diagram of IR Sensor:



Pin mapping:

ETS-IoT Kit	SENSOR
5V	VCC
GPA0	OUTPUT
GND	GND

Software Example:

```
# Application to connect a IR(Digital) sensor and print the status of Object presence.

import sys
sys.path.append('/home/pi/Adafruit-Raspberry-Pi-Python-Code-
legacy/Adafruit_MCP230xx')

from Adafruit_MCP230XX import Adafruit_MCP230XX
import time

mcp = Adafruit_MCP230XX(busnum = 1, address = 0x21, num_gpios = 16)
```

```
# Set pins 0, 1 and 2 to output (you can set pins 0..15 this way)
mcp.config(0, mcp.INPUT)
mcp.pullup(0, 1)


while (True):
   i = mcp.input(0)
   time.sleep(1)

   if i == 1:
      print "Obstacle detect"

   if i == 0:
      print "Obstacle not detect"
```

FLAME SENSOR:

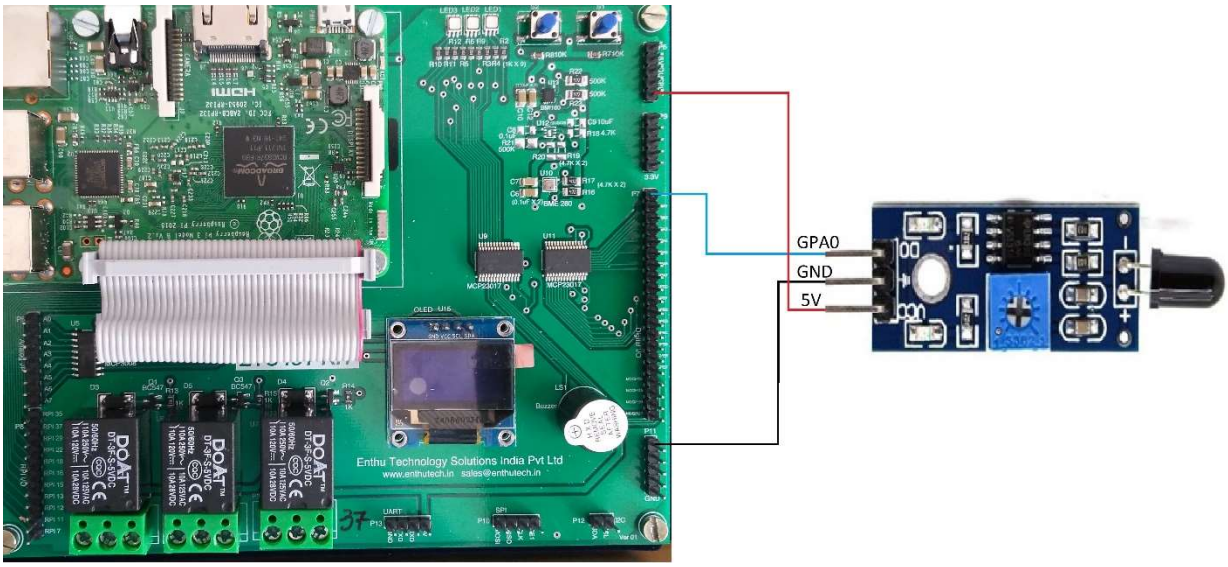
Flame Detection Sensor Module is sensitive to the flame, but also can detect ordinary light. Usually used as a flame alarm.

Detects a flame or a light source of a wavelength in the range of 760nm-1100 nm. Detection point of about 60 degrees, particularly sensitive to the flame spectrum. Sensitivity is adjustable, stable performance.

Applications of FLAME Sensors:

- Aircraft hangars, clothing dryers and high voltage equipment
- Gas fueled cookers and domestic heating systems
- Industrial heating and drying systems, and furnaces
- Industrial gas turbines and oil refineries
- Engine test facilities and engine rooms
- Generators and storage tanks.

Connection diagram of FLAME Sensor:



Pin mapping:

ETS-IoT Kit	SENSOR
5V	VCC
GPA0	DO
GND	GND

Software Example:

```
# Application to connect a FLAME(Digital) sensor and print the status of Fire detection.

import sys
sys.path.append('/home/pi/Adafruit-Raspberry-Pi-Python-Code-
legacy/Adafruit_MCP230xx')

from Adafruit_MCP230XX import Adafruit_MCP230XX
import time

mcp = Adafruit_MCP230XX(busnum = 1, address = 0x21, num_gpios = 16)
```

```
# Set pins 0, 1 and 2 to output (you can set pins 0..15 this way)
mcp.config(0, mcp.INPUT)
mcp.pullup(0, 1)


while (True):
   i = mcp.input(0)
   time.sleep(1)

   if i == 1:
      print "Fire detect"

   if i == 0:
      print "Fire not detect"
```

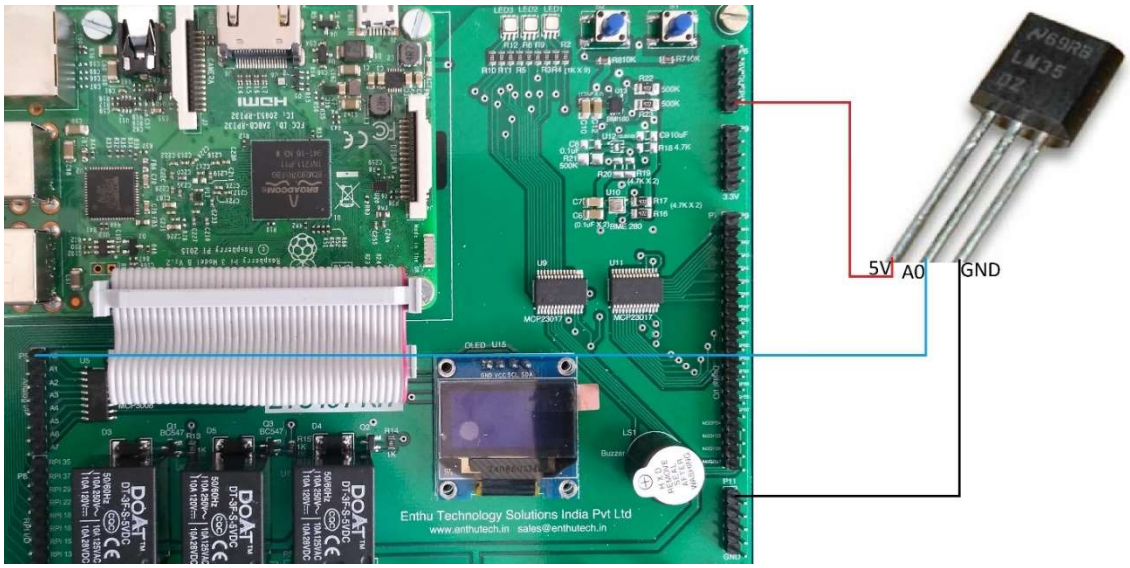
LM35 SENSOR:

LM35 is a precision IC temperature sensor with its output proportional to the temperature (in ^{o}C). The sensor circuitry is sealed and therefore it is not subjected to oxidation and other processes. With **LM35**, temperature can be measured more accurately than with a thermistor. It also possess low self heating and does not cause more than 0.1 ^{o}C temperature rise in still air.

Applications of LM35 Sensors:

- Power Supplies
- Battery Management
- HVAC
- Appliances

Connection diagram of LM35:



Pin mapping:

ETS-IoT Kit	SENSOR
5V	VCC
A0	AO
GND	GND

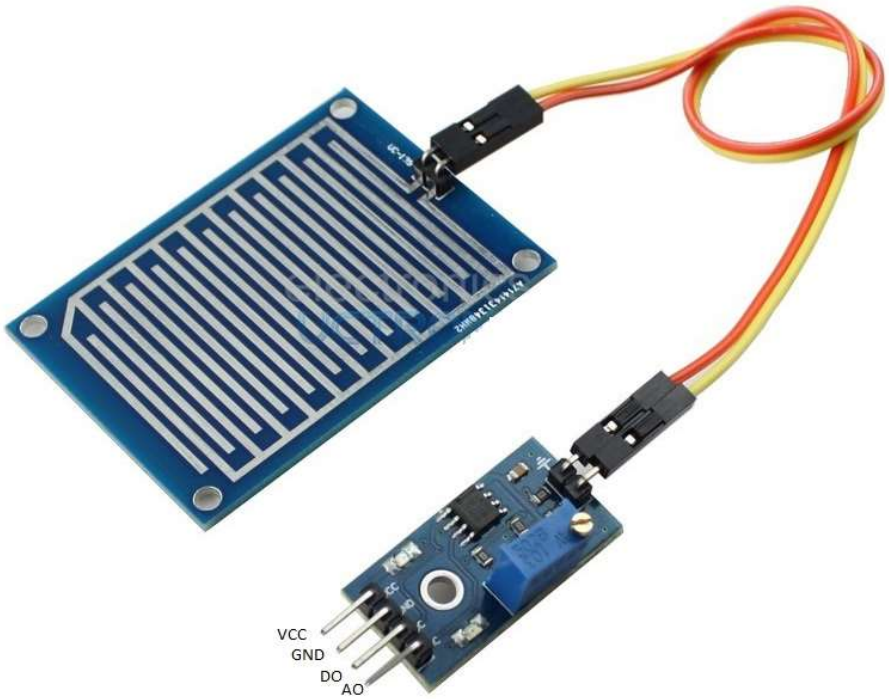
Software Example:

```
# Application to connect a LM35(Analog) sensor and print the Temperature.

import RPi.GPIO as GPIO
import spidev
import time

spi = spidev.SpiDev()
spi.open(0,0)


def ReadChannel(channel):
   adc = spi.xfer2([1,(8+channel)<<4,0])
   data= ((adc[1]&3) << 8) + adc[2]
```

```
    return data

def ConvertVolts(data,places):
    volts = (data * 3.3) / float(1023)
    volts = round(volts,places)
    return volts

def ConvertTemp(data,places):
    temp = ((data * 330)/float(1023))-5
    temp = round(temp,places)
    return temp

temp_channel = 0
delay = 3
while True:

    temp_level = ReadChannel(temp_channel)
    temp_volts = ConvertVolts(temp_level,2)
    temp       = ConvertTemp(temp_level,2)

    print "_____"
    print ("Temp : {} ({}V) {} deg C".format(temp_level,temp_volts,temp))
    time.sleep(delay)
```

RAIN SENSOR:

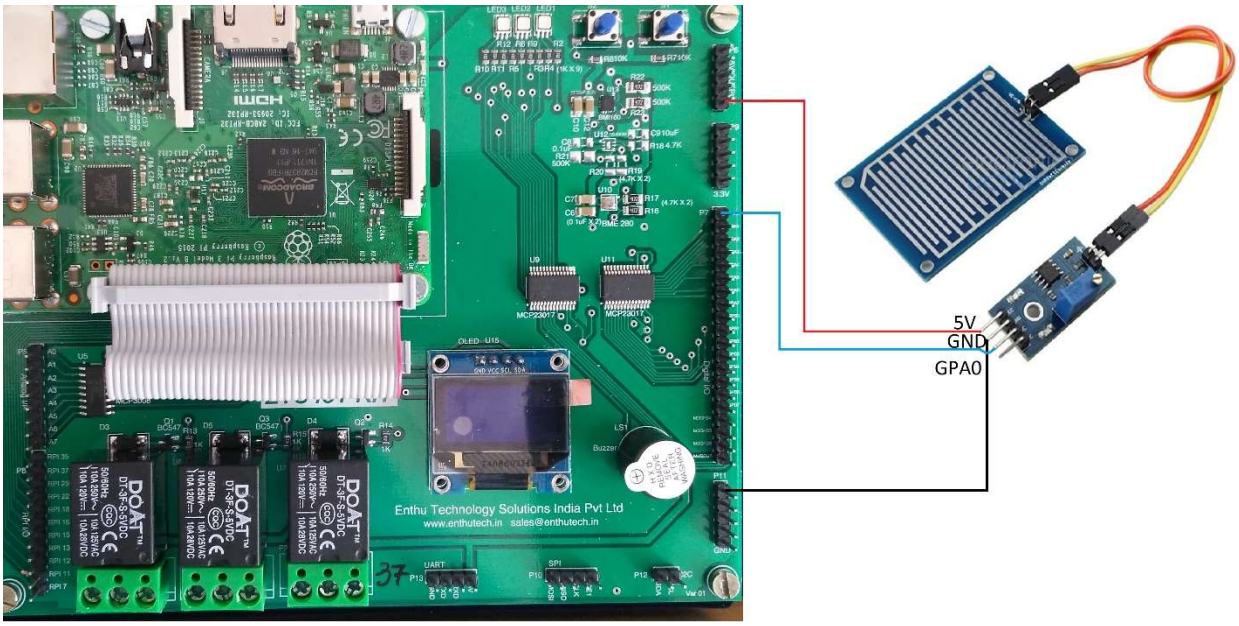
The sensor plate has 2 PCB tracks which routed parallelly surrounding it. These tracks are not connected, but when water/rain drop to the surface of plate and changes the resistance between the tracks because water/rain is conductive, this further changes the resistance between tracks and lower down the resistance. More water/rain touches the plate, resistance become lower.



Applications of RAIN Sensor:

- Rain detection

Connection diagram of RAIN Sensor:



Pin mapping:

ETS-IoT Kit	SENSOR
5V	VCC
GPA0	DO
GND	GND

Software Example:

```
# Application to connect a RAIN(Digital) sensor and print the Water/Rain availability.

import sys
sys.path.append('/home/pi/Adafruit-Raspberry-Pi-Python-Code-
legacy/Adafruit_MCP230xx')

from Adafruit_MCP230XX import Adafruit_MCP230XX
import time

mcp = Adafruit_MCP230XX(busnum = 1, address = 0x21, num_gpios = 16)
```

```
# Set pins 0, 1 and 2 to output (you can set pins 0..15 this way)
mcp.config(0, mcp.INPUT)
mcp.pullup(0, 1)


while (True):
   i = mcp.input(0)
   time.sleep(1)

   if i == 1:
      print "Rain detect"

   if i == 0:
      print "Rain not detect"
```

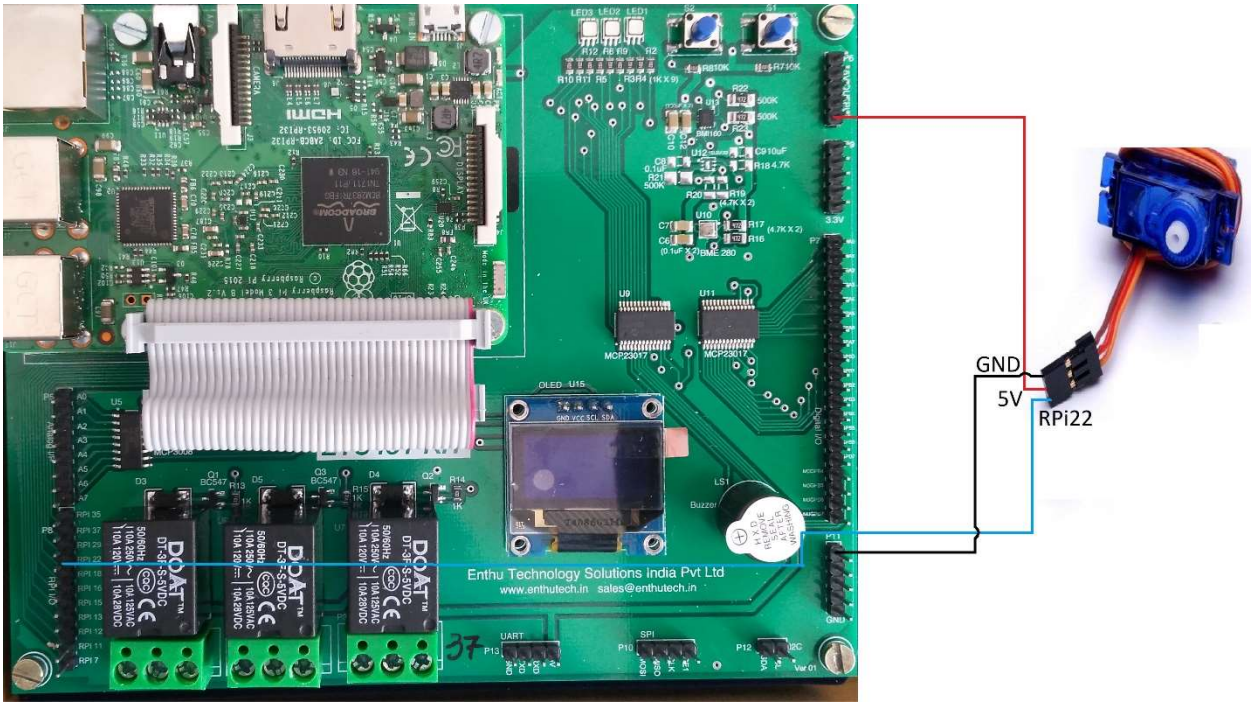
SERVO MOTOR:

The servo motor is most commonly used for high technology devices in the industrial application like automation technology. It is a self-contained electrical device that rotate parts of a machine with high efficiency and great precision. The output shaft of this motor can be moved to a particular angle. Servo motors are mainly used in home electronics, toys, cars, airplanes, etc.

Applications of Servo Motor:

- Robotics
- Security Cameras
- solar tracking system
- milling machines
- Textiles

Connection diagram of SERVO MOTOR:



Pin mapping:

ETS-IoT Kit	SENSOR
5V	VCC
RPi22	INPUT
GND	GND

Software Example:

```
# Application to connect a SERVO Motor and Rotate 180° mode of operation.

import RPi
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setup(22, GPIO.OUT)

pwm=GPIO.PWM(22,100)
pwm.start(5)

angle1=10
duty1= float(angle1)/10 + 2.5

angle2=160
duty2= float(angle2)/10 + 2.5

ck=0
while ck<=5:
    pwm.ChangeDutyCycle(duty1)
    time.sleep(0.8)
    pwm.ChangeDutyCycle(duty2)
    time.sleep(0.8)
    ck=ck+1
time.sleep(1)
GPIO.cleanup()
```

AIR SENSOR:

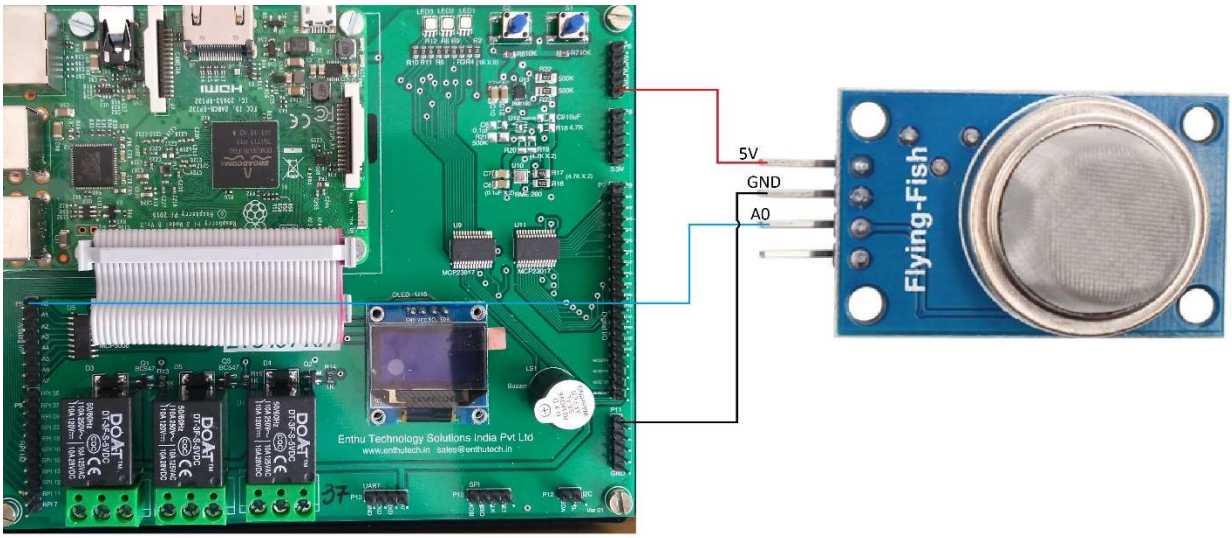
It is a hazardous gas detection apparatus for the family, the environment, suitable for ammonia, aromatic compounds, sulphur, benzene vapour, smoke and other gases harmful gas detection, gas-sensitive element test.

Air quality sensor is for detecting a wide range of gases, including NH3, NOx, alcohol, benzene, smoke and CO2. Ideal for use in office or factory with simple drive and monitoring circuit.

Applications of AIR Sensor:

- Domestic air pollution detector
- Industrial air pollution detector
- Portable air pollution detector

Connection diagram of AIR Sensor:



Pin mapping:

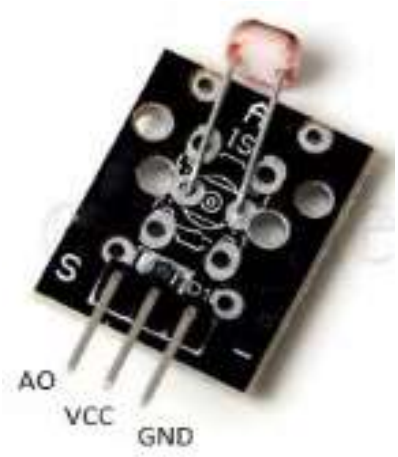
ETS-IoT Kit	SENSOR
5V	VCC
A0	AO
GND	GND

Software Example:

```
# Application to connect a AIR Quality(Analog) sensor and print the status of Air Quality.

import spidev
import time

spi = spidev.SpiDev()
spi.open(0,0)


def ReadChannel(channel):
  adc = spi.xfer2([1,(8+channel)<<4,0]) #as per datasheet
  data= ((adc[1]&3) << 8) + adc[2]
  return data

def ConvertVolts(data,places):
  volts = (data * 3.3) / float(1023)
  volts = round(volts,places)
  return volts


Air_channel = 0 ##connect with A0
delay = 3

while True:


  Air_level = ReadChannel(Air_channel)
  Air_volts = ConvertVolts(Air_level,2)

  print "_____"
  print ("Air Quality: {} ({}V)".format(Air_level, Air_volts))

  time.sleep(delay)
```

LIGHT DEPENDENT RESISTOR:

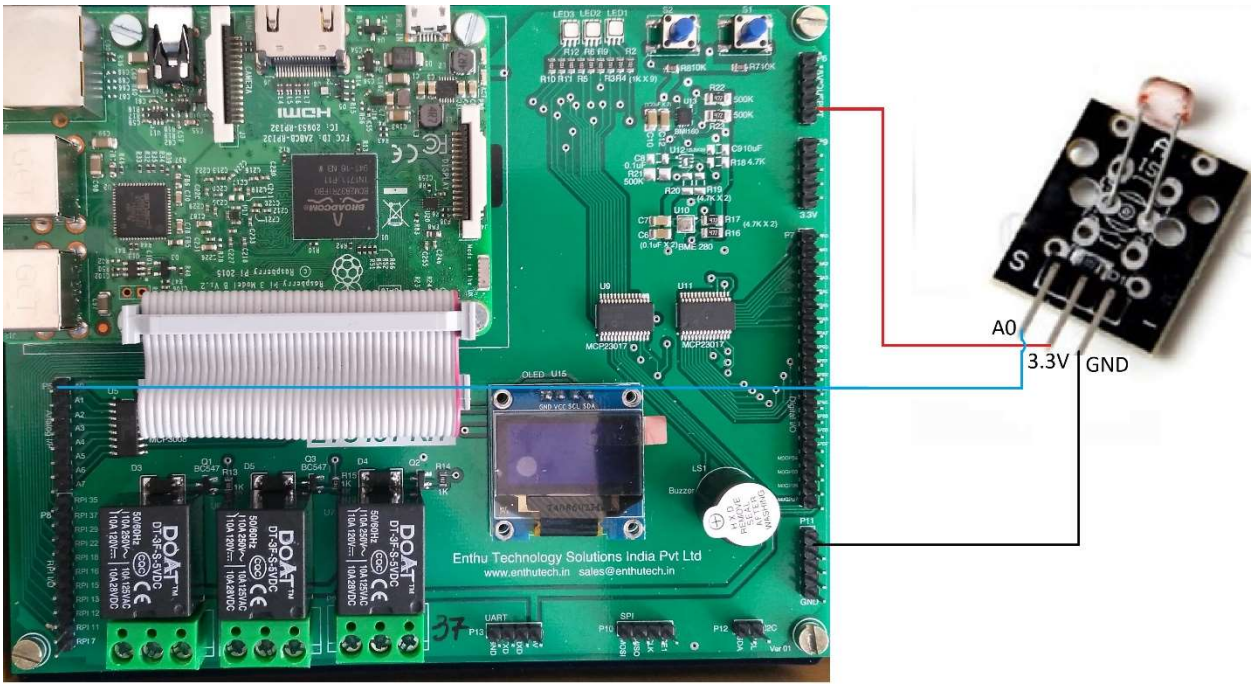
An LDR or light dependent resistor is also known as photo resistor, photocell, photoconductor. It is a one type of resistor whose resistance varies depending on the amount of light falling on its surface. When the light falls on the resistor, then the resistance changes.



Applications of AIR Sensor:

- Automatic Headlight Dimmer
- Night Light Control
- Street Light Control
- Camera Exposure Control

Connection diagram of LDR:



Pin mapping:

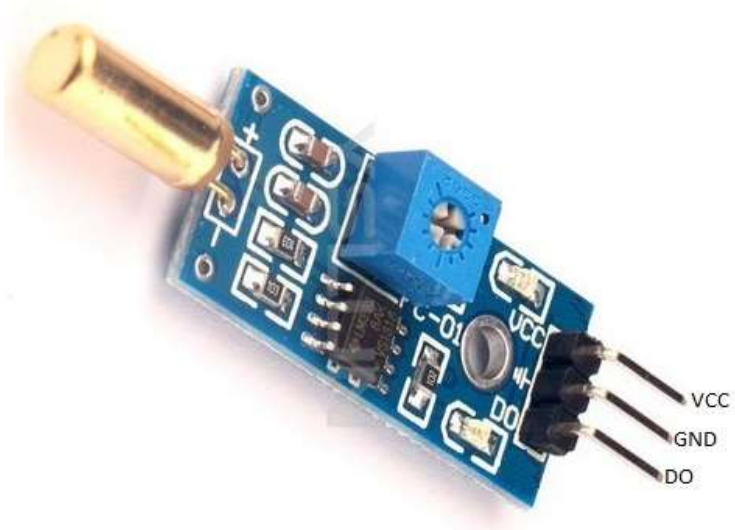
ETS-IoT Kit	SENSOR
5V	VCC
A0	AO
GND	GND

Software Example:

```
# Application to connect a LDR(Analog) sensor and print the value of Light resistence.

import spidev
import time

spi = spidev.SpiDev()
spi.open(0,0)


def ReadChannel(channel):
  adc = spi.xfer2([1,(8+channel)<<4,0]) #as per datasheet
  data= ((adc[1]&3) << 8) + adc[2]
  return data

def ConvertVolts(data,places):
  volts = (data * 3.3) / float(1023)
  volts = round(volts,places)
  return volts


Light_channel = 0 ##connect with A0
delay = 3

while True:


  Light_level = ReadChannel(Light_channel)
  Light_volts = ConvertVolts(Light_level,2)

  print "_____"
  print ("Light Value: {} ({}V)".format(Light_level, Light_volts))

  time.sleep(delay)
```

VIBRATION SENSOR:

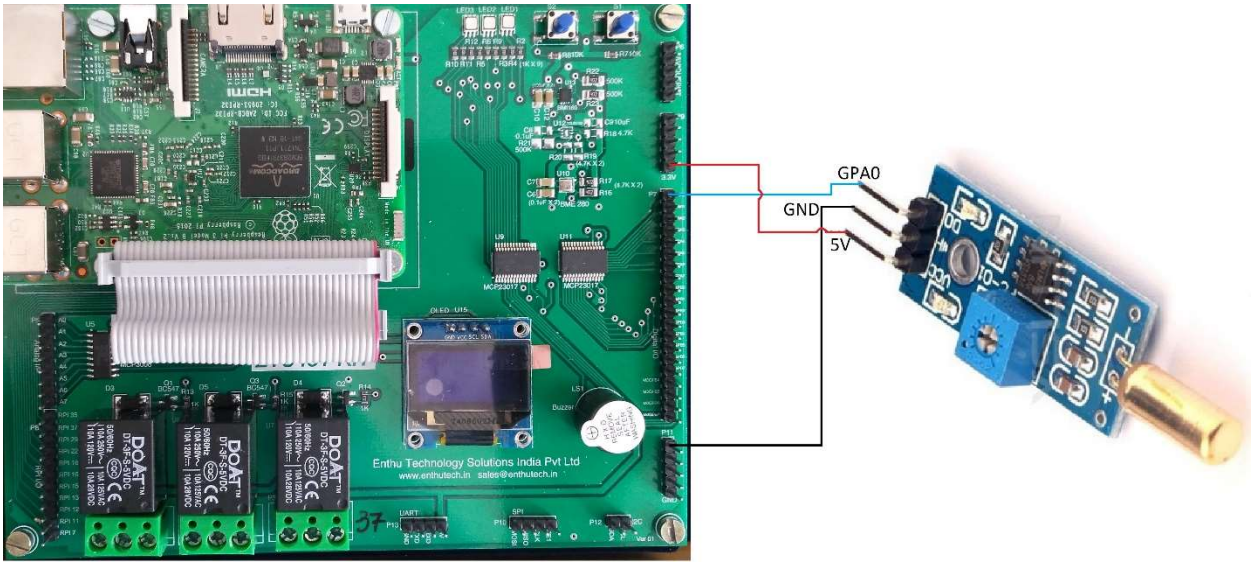
The Vibration module based on the vibration sensor SW-420 and Comparator LM393 to detect if there is any vibration that beyond the threshold. The threshold can be adjusted by the on-board potentiometer.



Applications of VIBRATION Sensor:

- Vibration detecting
- Burglary protection system

Connection diagram of VIBRATION Sensor:



Pin mapping:

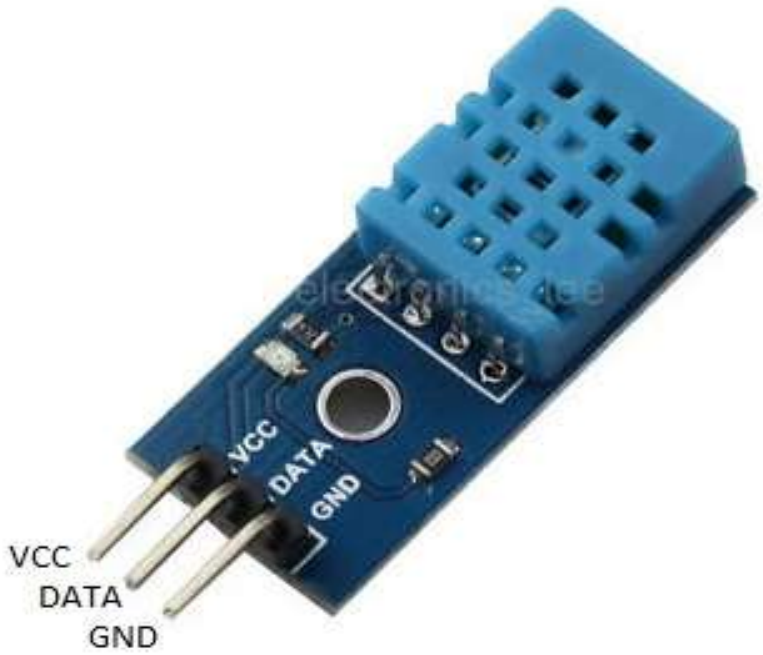
ETS-IoT Kit	SENSOR
5V	VCC
GPA0	DO
GND	GND

Software Example:

```
# Application to connect a VIBRATION(Digital) sensor and print the status of Vibration happen.

import RPi.GPIO as GPIO
import time

#GPIO SETUP
channel = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(channel, GPIO.IN)

def callback(channel):
    if GPIO.input(channel):
        print "Movement Detected!"
    else:
        print "Movement Detected!"

GPIO.add_event_detect(channel, GPIO.BOTH, bouncetime=300)  # let us know when the pin goes HIGH or LOW
GPIO.add_event_callback(channel, callback)  # assign function to GPIO PIN, Run function on change

# infinite loop
while True:
    time.sleep(1)
```

DHT11 SENSOR:

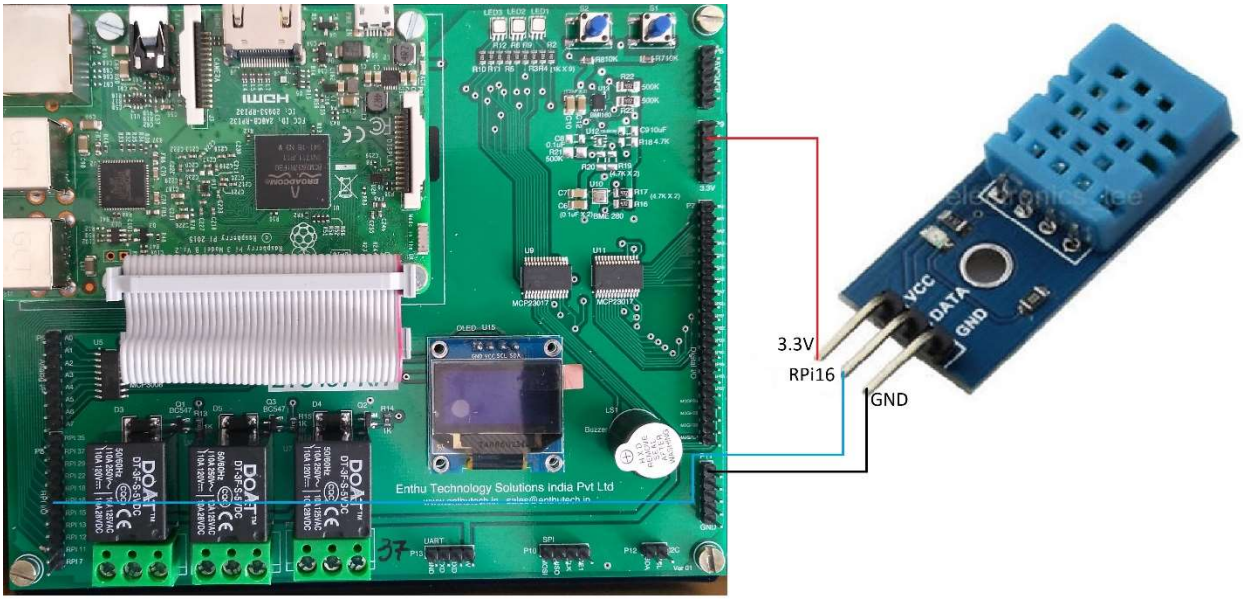
DHT11 digital temperature and humidity sensor is a composite Sensor contains a calibrated digital signal output of the temperature and humidity.



Applications of DHT11 Sensor:

- HVAC
- testing and inspection equipment
- consumer goods
- automotive
- automatic control
- data loggers
- weather stations
- home appliances

Connection diagram of DHT11:



Pin mapping:

ETS-IoT Kit	SENSOR
5V	VCC
FPi16	DATA
GND	GND

Before running the python script do the following:

In the terminal type:
sudo apt-get install git-core

Download the Adafruit DHT11 library. In the terminal type:
git clone https://github.com/adafruit/Adafruit_Python_DHT.git

Navigate to to Adafruit_Python_DHT directory (folder), in the terminal type:
cd Adafruit_Python_DHT

To install the library, in the terminal type:
sudo python setup.py install

Software Example:

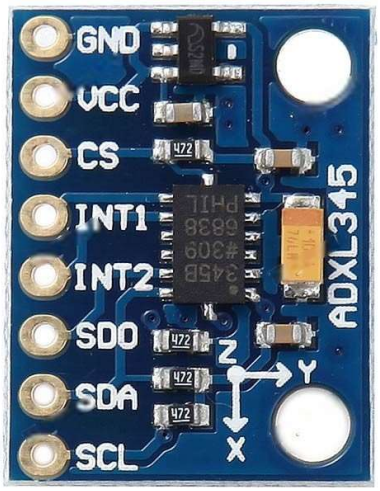
```
# Application to connect a DHT11(Digital) sensor and print the Temperature and Humidity.

import Adafruit_DHT

sensor = Adafruit_DHT.DHT11

pin = 23

humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)

while True:
   if humidity is not None and temperature is not None:
      print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature, humidity))
   else:
      print('Failed to get reading. Try again!')
```

Reference Link: *http://invent.module143.com/daskal_tutorial/raspberry-pi-3-gpio-dht11-digital-temperature-humidity-sensor/*

ADXL345 SENSOR:

The ADXL345 digital 3-axis accelerometer module is ideal for motion and acceleration sensing applications.

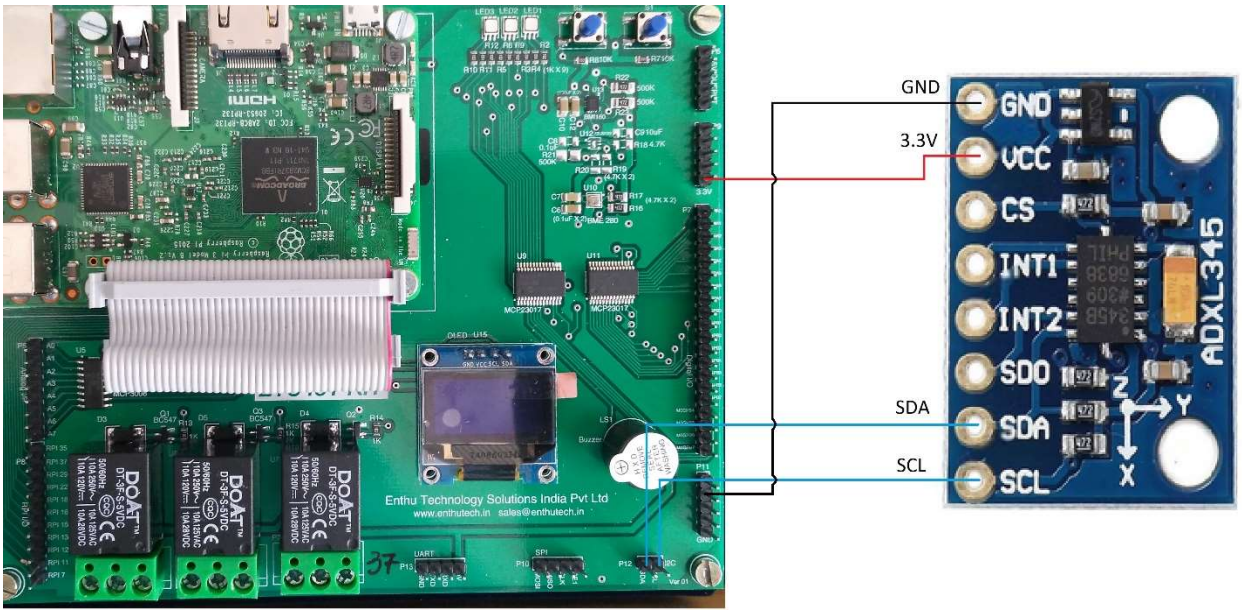
This module contains an on-board ADXL345 accelerometer with digital data output formatted as 16-bit twos complement and accessible through either an I2C or SPI interface.



Applications of ADXL345 Sensor:

- Handsets
- Medical instrumentation
- Gaming and pointing devices
- Industrial instrumentation
- Personal navigation devices
- Hard disk drive (HDD) protection
- Fitness equipment

Connection diagram of ADXL345:



Pin mapping:

ETS-IoT Kit	SENSOR
3.3V	VCC
SDA	SDA
SCL	SCL
GND	GND

Download the Adafruit library. In the terminal type:
git clone https://github.com/pimoroni/adxl345-python.git

Software Example:

```
# Application to connect a ADXL345(Digital) sensor and print the acceleration of X, Y and Z.

from adxl345 import ADXL345
import time

adxl345 = ADXL345()
while True:
   axes = adxl345.getAxes(True)

   print "ADXL345 on address 0x%x:" % (adxl345.address)
   print "   x = %.3fG" % ( axes['x'] )
   print "   y = %.3fG" % ( axes['y'] )
   print "   z = %.3fG" % ( axes['z'] )
   time.sleep(0.5)
```