Sensors

1. Temperature & Humidity Sensor (DHT11) :

DHT11 is a low-cost digital sensor for sensing temperature and humidity. This sensor can be easily interfaced with any micro-controller such as Arduino, Raspberry Pi etc… to measure humidity and temperature instantaneously.

Working :

DHT11 sensor consists of a capacitive humidity sensing element and a thermistor for sensing temperature. The humidity sensing capacitor has two electrodes with a moisture holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels. The IC measure, process this changed resistance values and change them into digital form.

For measuring temperature this sensor uses a Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with increase in temperature. To get larger resistance value even for the smallest change in temperature, this sensor is usually made up of semiconductor ceramics or polymers.

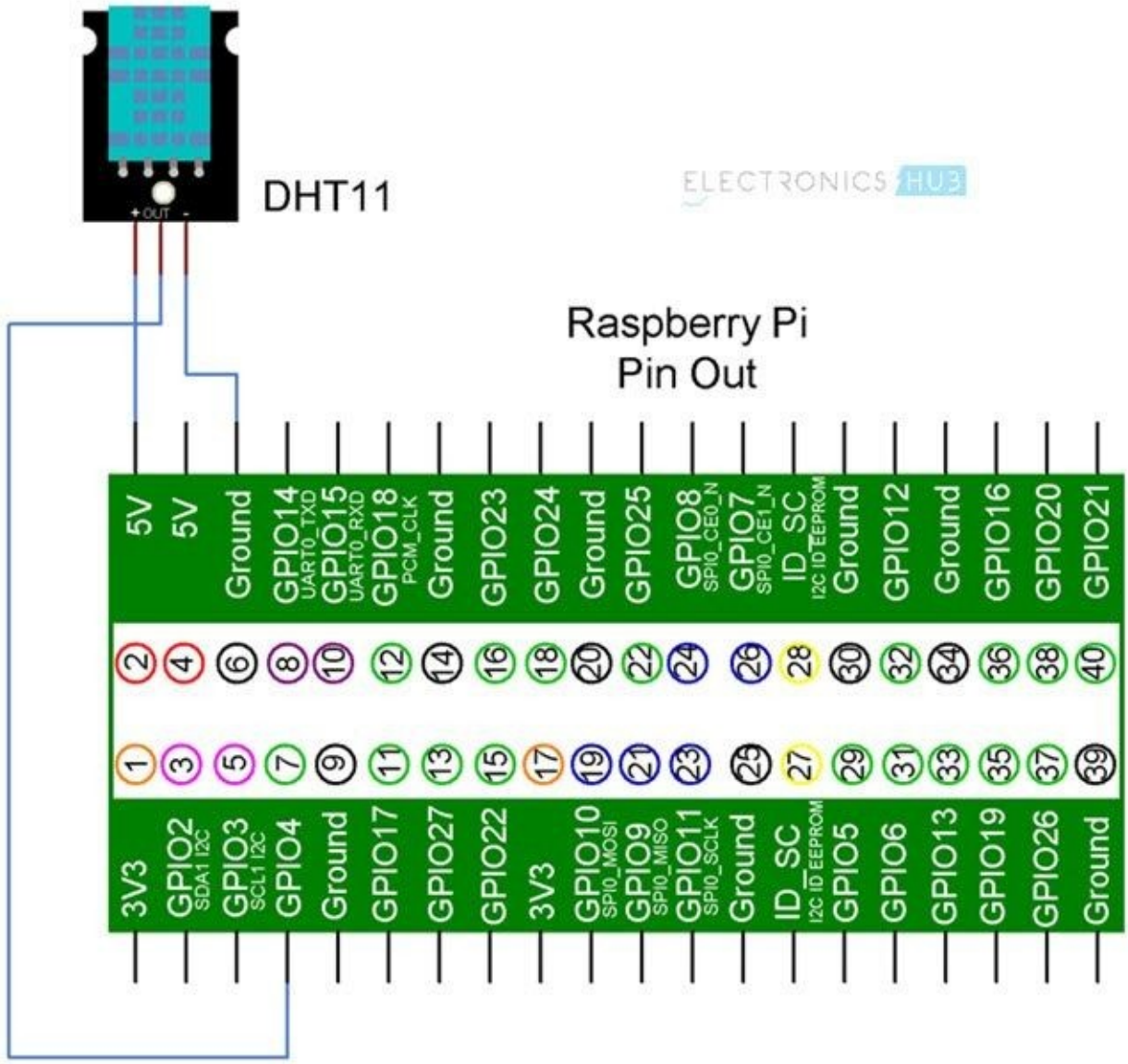
Features :

- The temperature range of DHT11 is from 0 to 50 degree Celsius with a 2-degree accuracy.
- Humidity range of this sensor is from 20 to 80% with 5% accuracy.
- The sampling rate of this sensor is 1Hz .i.e. it gives one reading for every second.
- DHT11 is small in size with operating voltage from 3 to 5 volts. The maximum current used while measuring is 2.5mA.
- DHT11 sensor has four pins- VCC, GND, Data Pin and a not connected pin.

Code :

```
import sys

import Adafruit_DHT

import time



while True:

        humidity, temperature = Adafruit_DHT.read_retry(11, 4)

        print 'Temp: {0:0.1f} C  Humidity: {1:0.1f} %'.format(temperature, humidity)

        time.sleep(1)
```

Interfacing with Pi:



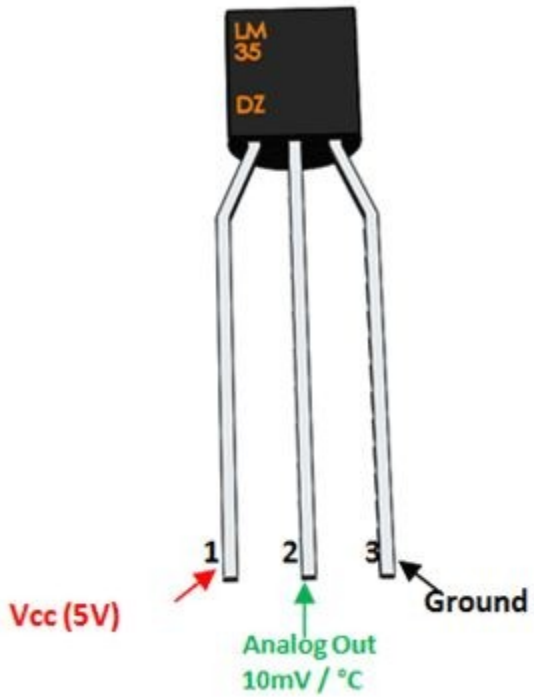
Uses :

- This sensor is used in various applications such as measuring humidity and temperature values in heating, ventilation and air conditioning systems.

- Weather stations also use these sensors to predict weather conditions.
- The humidity sensor is used as a preventive measure in homes where people are affected by humidity.
- Offices, cars, museums, greenhouses and industries use this sensor for measuring humidity values and as a safety measure.

2. Temperature sensor (LM35)

LM35 is a precession Integrated circuit Temperature sensor, whose output voltage varies, based on the temperature around it. It is a small and cheap IC which can be used to measure temperature anywhere between -55°C to 150°C. It can easily be interfaced with any Microcontroller that has ADC function or any development platform like Arduino.



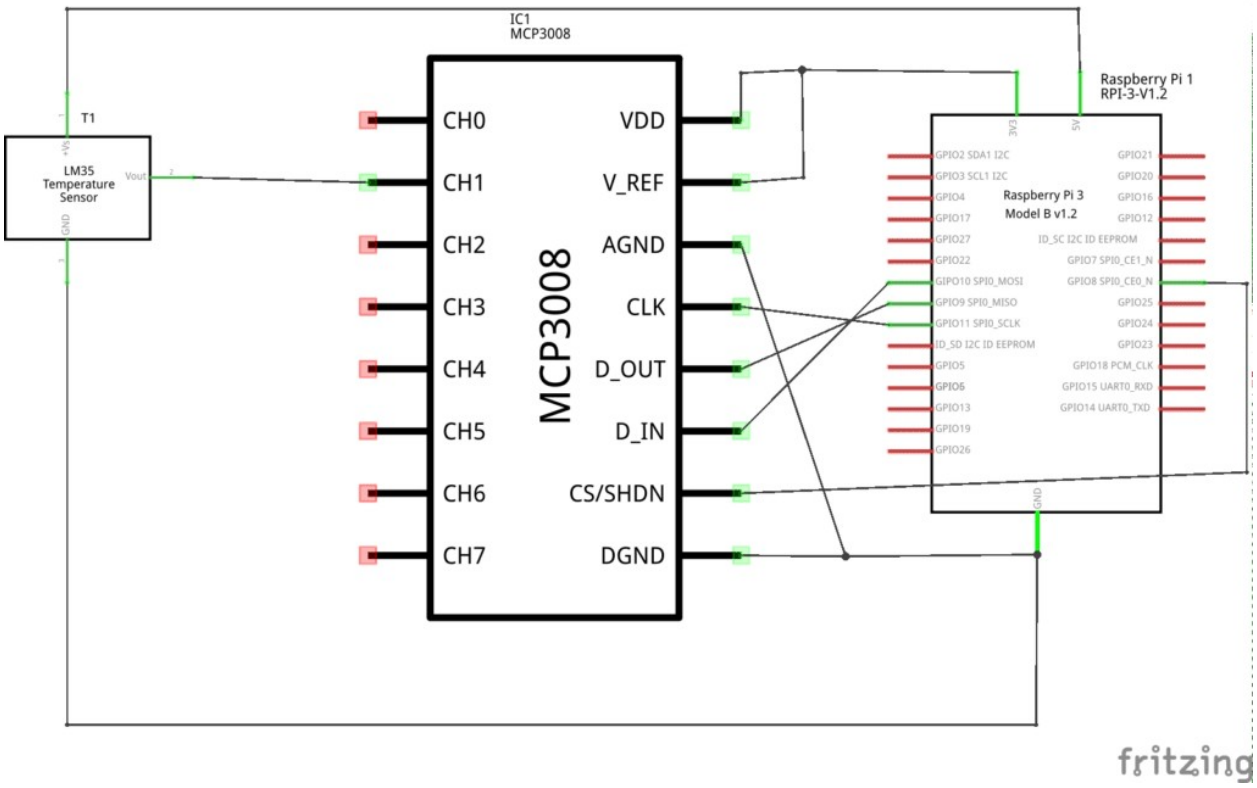
Features:

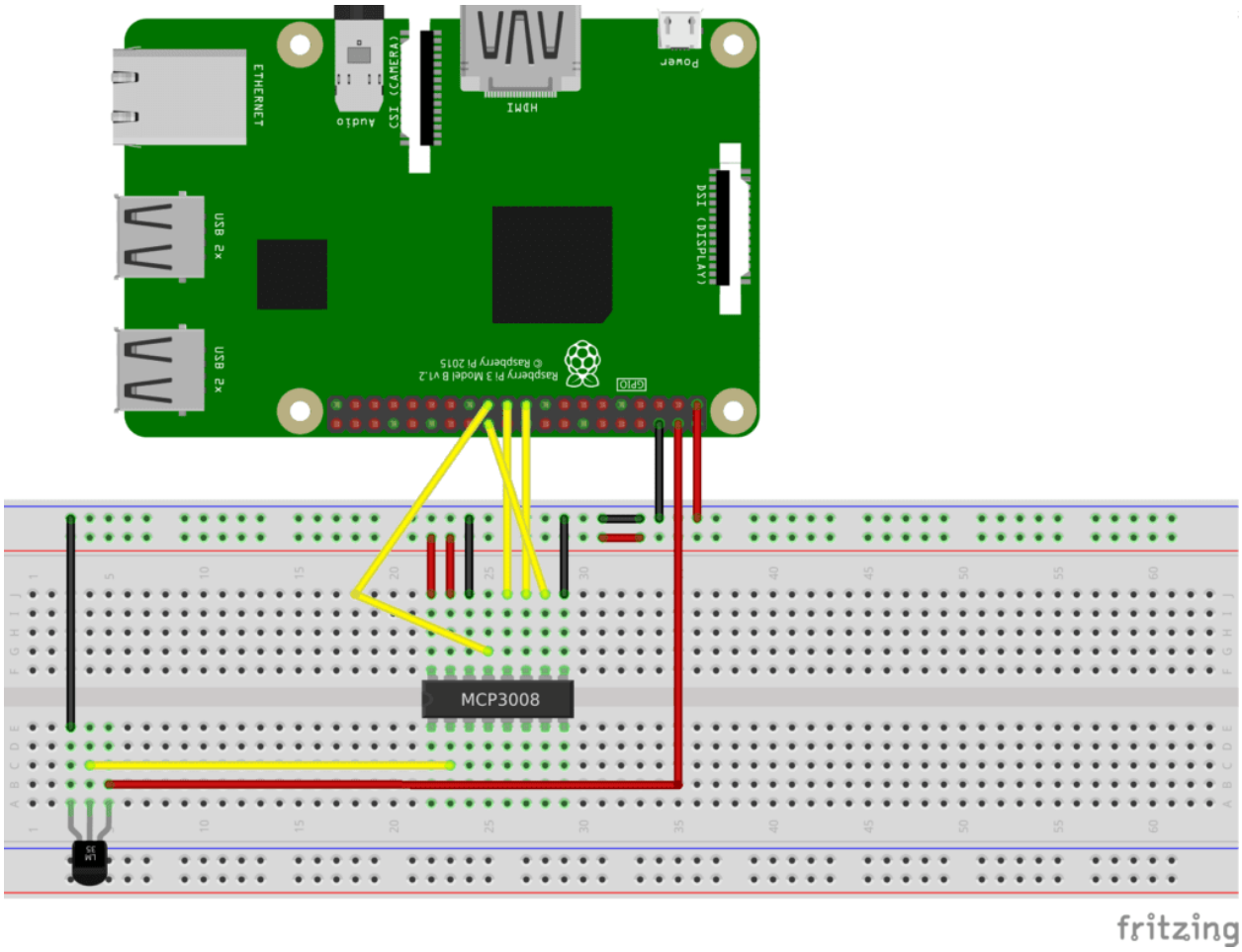
- Minimum and Maximum Input Voltage is 35V and -2V respectively. Typically 5V.
- Can measure temperature ranging from -55°C to 150°C
- Output voltage is directly proportional (Linear) to temperature (i.e.) there will be a rise of 10mV (0.01V) for every 1°C rise in temperature.
- ±0.5°C Accuracy
- Drain current is less than 60uA
- Low cost temperature sensor
- Small and hence suitable for remote applications
- The output signal through LM35 is analog.

Interfacing with Pi:

The output of LM35 is an analog signal and the Raspberry PI does not have any analog input pin so we need an analog to digital signal converter in between.

MCP3008 is a chip designed to convert analog signal in to digital data can be used in between.


IC1
MCP3008
T1
LM35
Temperature
Sensor
Vout
GND
CH0
CH1
CH2
CH3
CH4
CH5
CH6
CH7
MCP3008
VDD
V_REF
AGND
CLK
D_OUT
D_IN
CS/SHDN
DGND
Raspberry Pi 1
RPI-3-V1.2
Raspberry Pi 3
Model B v1.2
fritzing



Source Code :

```
import spidev

from time import sleep




# First open up SPI bus
```

```
spi = spidev.SpiDev()

spi.open(0, 0)


# Initialize what sensor is where

tempChannel = 1

sleepTime = 1



def getReading(channel):

        # First pull the raw data from the chip

        rawData = spi.xfer([1, (8 + channel) &amp;lt;&amp;lt; 4, 0])

        # Process the raw data into something we understand.

        processedData = ((rawData[1] &amp;amp; 3) &amp;lt;&amp;lt; 8) + rawData[2]

        return processedData



def convertVoltage(bitValue, decimalPlaces=2):

        voltage = (bitValue * 3.3) / float(1023)

        voltage = round(voltage, decimalPlaces)
```

```
        return voltage



def convertTemp(bitValue, decimalPlaces=2):
        # Converts to degrees Celsius
        temperature = ((bitValue * 330) / float(1023))
        # 3300 mV / (10 mV/deg C) = 330
        temperature = round(temperature, decimalPlaces)
        return temperature

while (1):
        tempData = getReading(tempChannel)
        tempVoltage = convertVoltage(tempData)
        temperature = convertTemp(tempData)
        print ("Temp bitValue = {}; Voltage = {} V; Temp = {} C".format
        (tempData, tempVoltage, temperature))
        sleep(sleepTime)
```

3.PIR Sensor

Overview

Passive Infrared Sensors, often referred to as PIR Sensors (also IR Motion Sensors and Pyroelectric Sensors), are Motions Detectors that basically detects the changes in Infrared Radiations emitted by a person.

Every living and non-living thing which has a temperature greater than absolute zero will emit infrared radiations. Since the emitted energy is in the form of infrared radiation, whose wavelength is greater than that of our visible light, we humans cannot see those radiations.

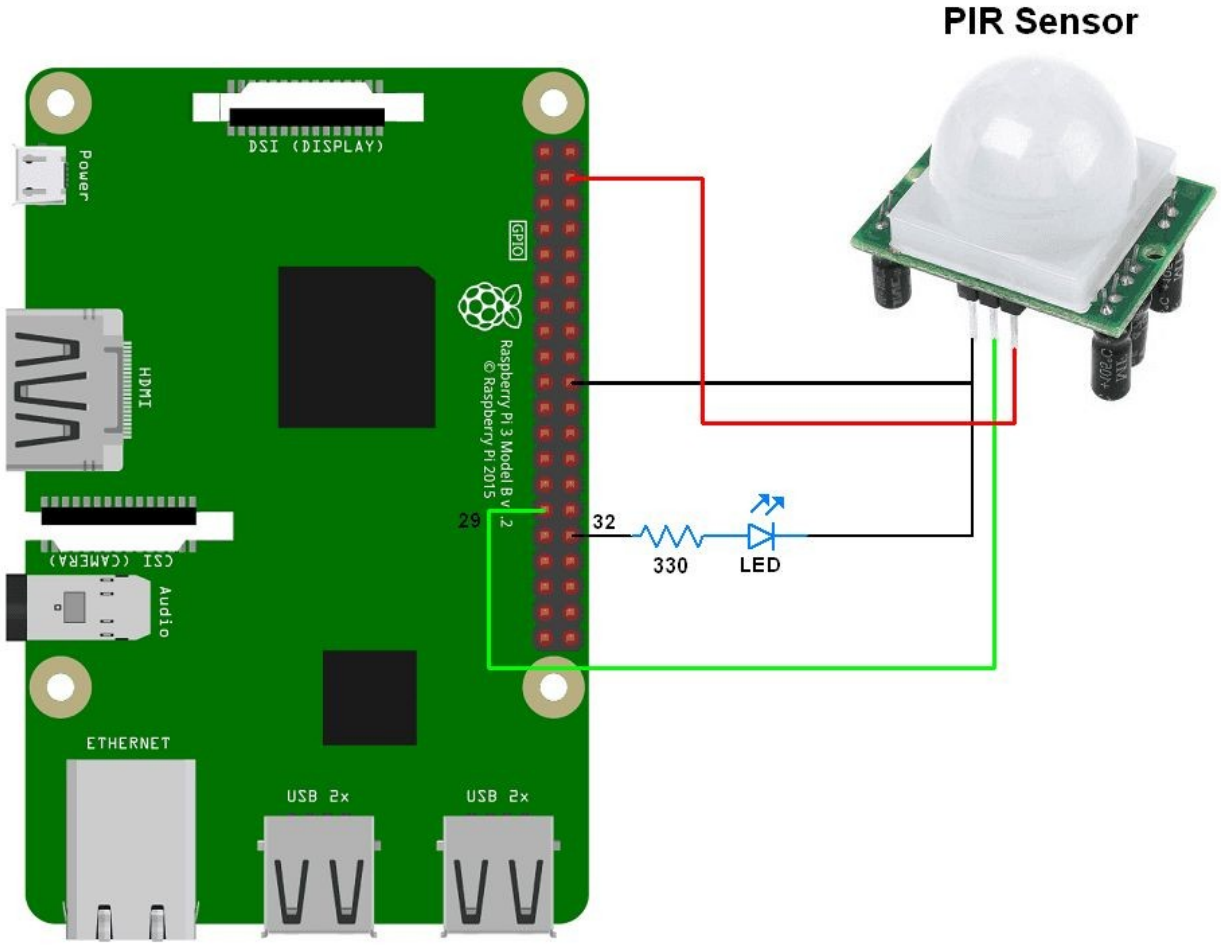
But PIR Sensors are built to detect those infrared radiations. Hence, they are employed in a variety of applications like Motion Detectors, Security Systems, Intruder Alert and so forth.

The term “Passive” in the PIR Sensor means that the sensor will not emit any infrared energy but rather detects infrared radiations emitted by other objects. This is in contra**st to active sensors, which perform both the actions (emitting and detection).**

Components Required

- **Raspberry Pi 3 Model B**
- **PIR Sensor**
- **5V Buzzer**
- **Connecting Wires**
- **Mini Breadboard**
- **Power Supply**
- **Computer**

Interfacing with 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()
```

4.IR Sensor:

Overview

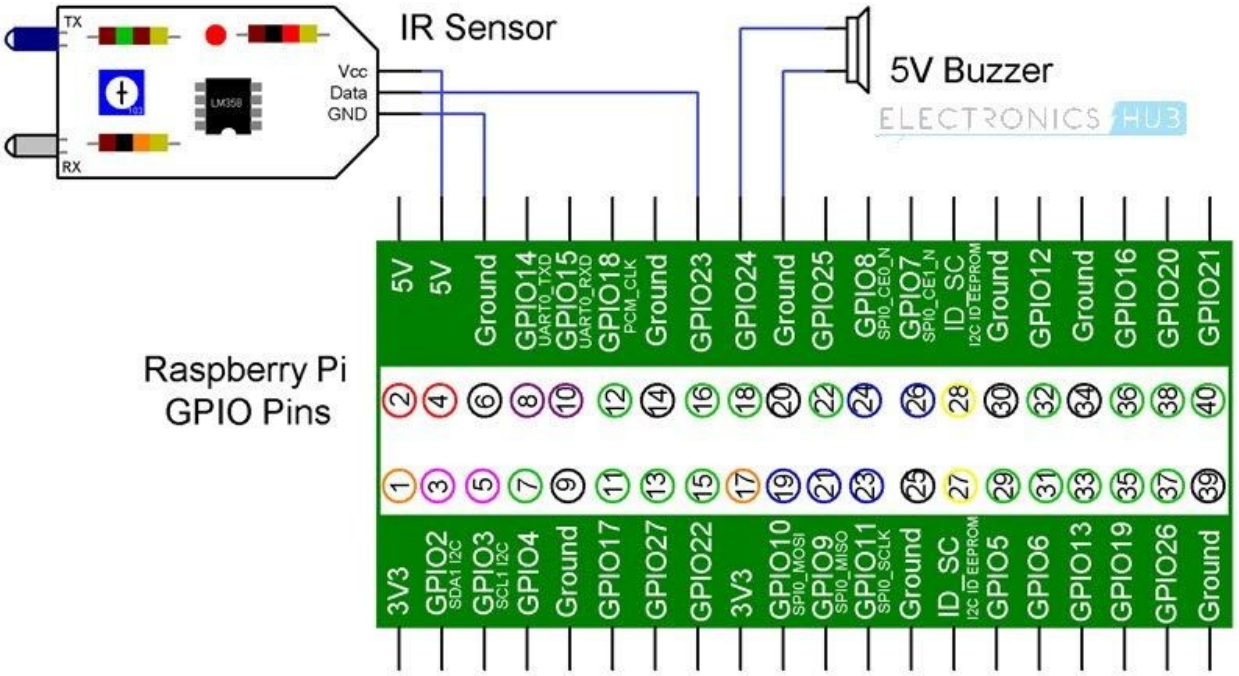
Infrared Sensors or IR Sensors are one of the frequently used sensor modules by electronics hobbyists and makers. They are often used as Obstacle Detecting Sensors or Proximity Sensors.

IR Sensors are also used in Contactless Digital Tachometers. Some of the other applications where IR Sensors are implemented are Line Follower Robots, Obstacle Avoiding Robots, Edge Avoiding Robots and many more.

Features:

- ATmega328P microcontroller
- Input voltage – 7-12V
- 14 Digital I/O Pins (6 PWM outputs)
- 6 Analog Inputs
- 32k Flash Memory
- 16Mhz Clock Speed

Interfacing with 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 "IR Sensor Ready....."
print " "
try:
  while True:
    if GPIO.input(sensor):
      GPIO.output(buzzer,True)
      print "Object Detected"
```

```
        while GPIO.input(sensor):
            time.sleep(0.2)
    else:
        GPIO.output(buzzer,False)


except KeyboardInterrupt:
    GPIO.cleanup()
```

5.Digital Light Sensor(BH1750):

Uses:

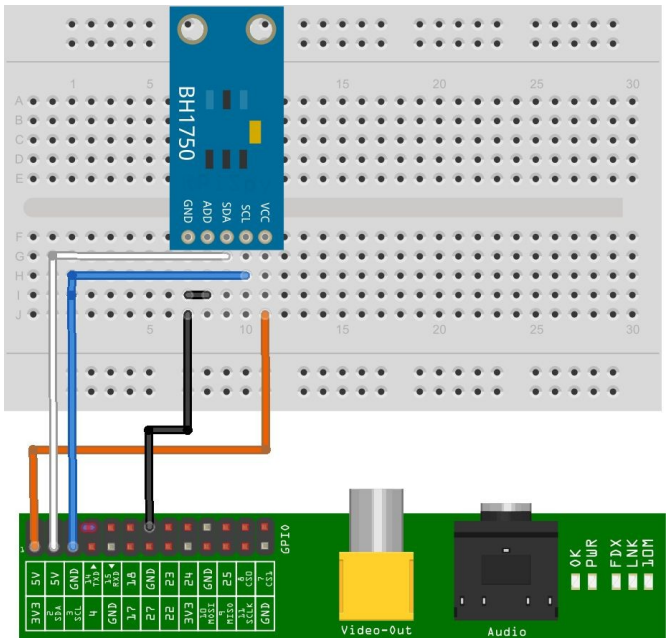
- It is used in automatic brightness adjustment in mobiles/LCD displays.
- Turning car headlights on/off depending on the environment

Features:

- Power Supply: 2.4V-3.6V (typically 3.0V)
- Highly responsive near to human eye.
- Very small effect of IR radiation
- Less current consumption: 0.12mA

Interfacing with Pi:

The following diagram shows the physical wiring between BH1750 and the Pi.



- Wire the GND pin of the BH1750 to Physical Pin 6 (GND) on the Raspberry Pi.
- Wire the VCC pin of the BH1750 to Physical Pin 1 (3v3) on the Raspberry Pi.
- Wire the SDA pin of the Bh1750 to Physical Pin 3 (SDA) on the Raspberry Pi.
- Wire the SCL pin of the Bh1750 to Physical Pin 5 (SCL) on the Raspberry Pi.

Sample Code:

```
import time
import smbus
bus = smbus.SMBus(0) #(256MB)
#bus = smbus.SMBus(1) #(512MB)
addr = 0x23 # i2c adress
while True:
  data = bus.read_i2c_block_data(addr,0x11)
  print "Luminosity " + str((data[1] + (256 * data[0])) / 1.2) + "lx"
```

```
time.sleep(0.5)
```

6. Accelerometer ADXL335:

Uses:

- Used for tilt screening applications(e.g. Racing games in smartphones)
- Used in construction working machines like drilling, driving piles and demolition.
- Used in disk drive protection, Sports and health devices.

Working:

It is a three axis analog accelerometer IC, which reads x, y and z acceleration as analog voltages. By measuring the amount of acceleration due to gravity, an accelerometer can figure out the angle it is tilted at with respect to the earth. By sensing the amount of dynamic acceleration, the accelerometer can find out how fast and in what direction the device is moving.

Features:

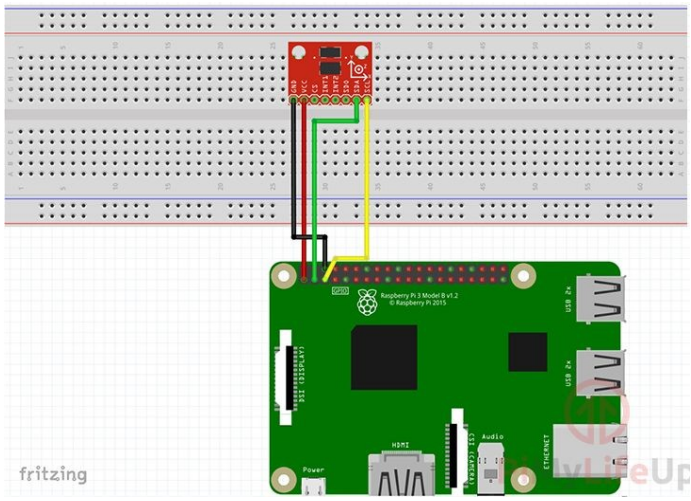
- 3V-6V DC Supply Voltage
- Onboard LDO Voltage regulator
- Can be interface with 3V3 or 5V Microcontroller.
- Free-Fall Detection
- Ultra Low Power: 40uA in measurement mode, 0.1uA in standby@ 2.5V
- Analog output

Interfacing with Pi:

The following diagram shows how the physical wiring between ADXL335 and Pi.

- Wire the GND pin of the Accelerometer to Physical Pin 6 (GND) on the Raspberry Pi.

- Wire the VCC pin of the Accelerometer to Physical Pin 1 (3v3) on the Raspberry Pi.
- Wire the SDA pin of the Accelerometer to Physical Pin 3 (SDA) on the Raspberry Pi.
- Wire the SCL pin of the Accelerometer to Physical Pin 5 (SCL) on the Raspberry Pi



Reading Information from the Sensor(Python Script):

- Install and import module “board” , which specifies what pins are available on a device.
- Install and import module “busio”, which contains a variety of libraries to handle different serial protocols.
- Finally install and import module “adafruit_adxl34x” which contains the required code to read the sensor ifo.

Calibration :

The calibration is done by determining the sensor outputs for each axis when it is precisely aligned with the axis of gravitational pull.

Sample Code:

```
import time
import board
import busio
import adafruit_adxl34x
```

```
i2c = busio.I2C(board.SCL, board.SDA)
accelerometer = adafruit_adxl34x.ADXL345(i2c)

while True:
        print("%f %f %f"%accelerometer.acceleration)
        time.sleep(1)
```