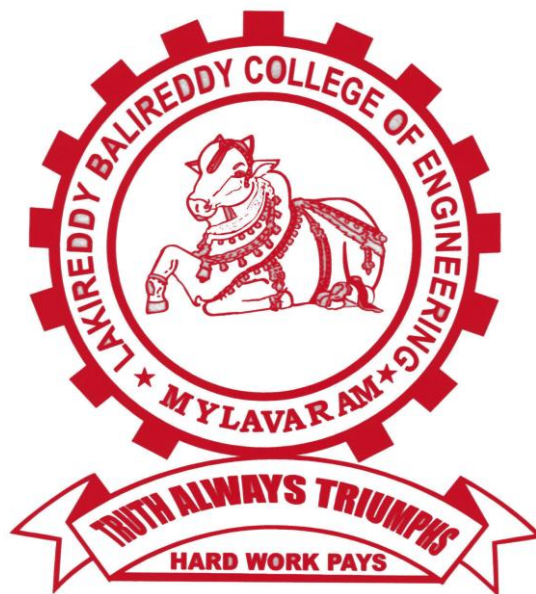


INTERNET OF THINGS



LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING
(AUTONOMOUS)

L.B.Reddy Nagar, Mylavaram – 521 230.

Affiliated to JNTUK, Kakinada & Approved by AICTE, New Delhi

NAAC Accredited with “A” grade, Accredited by NBA

Certified by ISO 9001-2008

1. Course Code : 20EC30
2. Course Title : IOT LAB Hands-on Laboratory Sessions
3. Core / Elective : Core
4. Pre-requisite : EMI, MPMC, Python Programming.
5. Year in which offered : IV Year B.Tech, ECE-VII SEM
6. No. of weeks of instruction : 17
7. No. of hours per week : 3 periods

Pre requisite: EMI, MPMC, Python Programming.

Course Educational Objective (CEO):

In this course, student will learn about basics of IoT and procedure to develop prototypes for engineering applications.

Course Outcomes (COs): At the end of this course, students will be able to:

CO1	:	Understand the programming concepts of IOT. (Understand – L2)
CO2	:	Develop real time applications using Internet of Things. (Apply – L3)
CO3	:	Demonstrate the integration of sensors with IOT. (Understand – L2)
CO4	:	Adapt effective Communication, presentation and report writing skills (Apply – L3)

COURSE ARTICULATION MATRIX:

Cos	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1															
CO2															
CO3															

Note: 1- Slight (Low), **2** – Moderate (Medium), **3** - Substantial (High), no correlation ‘-’

Prescribed Syllabus:**UNIT – I: IoT Basics:**

IoT, Frame work, Architectural View, Technology, Sources, M2M communication, Sensors, Participatory sensing, RFID, Wireless sensor network elements

UNIT – II: IoT Applications:

Prototyping embedded devices for M2M and IoT, M2M and IoT case studies.

TEXTBOOK:

1. Raj Kamal, Internet of Things - Architecture and Design Principles, McGraw Hill Publication, 2017.
2. Zach Shelby, Carsten Bormann: “The Wireless Embedded Internet”, Wiley, 1st Edition.

REFERENCES:

1. Arshdeep Bahga and Vijay Madisetti, Internet of Things – A Hands-on Approach, University Press, 2015
2. Reema Thareja, “Python Programming using Problem Solving Approach”, Oxford Press.

HANDS – ON LABORATORY SESSIONS

1. Interfacing LED. DHT11- Temperature and, humidity sensor using Arduino
2. Interfacing Ultrasonic sensor and PIR sensor using Arduino
3. Design of Traffic Light Simulator using Arduino
4. Design of Water flow detection using an Arduino board
5. Interfacing of LED, Push button with Raspberry Pi and Python Program
6. Design of Motion Sensor Alarm using PIR Sensor
7. Interfacing DHT11-Temperature and Humidity Sensor with Raspberry Pi
8. Interfacing DS18B20 Temperature Sensor with Raspberry Pi
9. Implementation of DC Motor and Stepper Motor Control with Raspberry Pi
10. Raspberry Pi based Smart Phone Controlled Home Automation
11. Smart Traffic light Controller
12. Smart Health Monitoring System

10. List of equipment and soft ware tools used:

INDEX				
Ex. No	DATE	NAME OF THE EXPERIMENT	Marks	Signature
1.				
2.				
3.				
4.				
5.				
6.				
7.				
8.				
9.				
10.				
11.				
12.				
13.				
14.				
15.				

Day to Day Evolution Marks Obtained	
Number of Experiments Done	
Average Marks Obtained	

Fundamentals of IoT Using Arduino

BLINKING LED USING ARDUINO

EXPT. NO :

DATE :

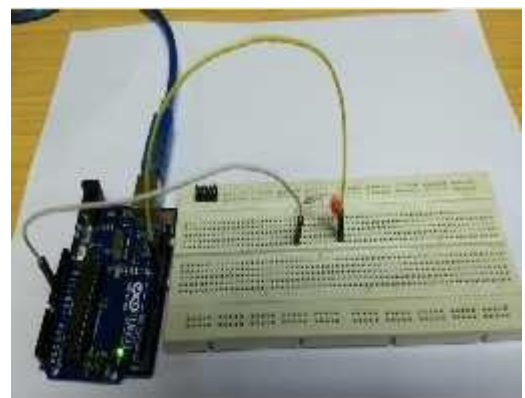
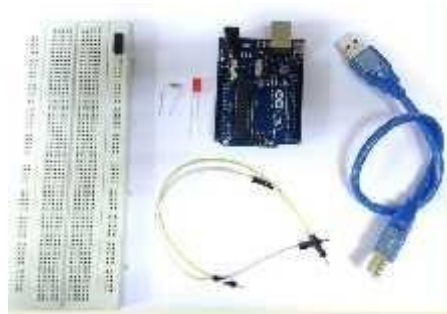
AIM: Blinking LED

Components Required:

1. Arduino controller board,
2. USB connector,
3. Bread board, LED, 1.4Kohm resistor,
4. Connecting wires,
5. Arduino IDE

Connection Procedure:

1. Connect the LED to the Arduino using the Bread board and the connecting wires.
2. Connect the Arduino board to the PC using the USB connector;
3. Select the board type and port.
4. Write the sketch in the editor, verify and upload.
5. Connect the positive terminal of the LED to digital pin 12 and the negative terminal to the ground pin (GND) of Arduino Board.



Programme: Sketch

```
void setup()
{
  pinMode (12, OUTPUT); // set the pin mode
}

void loop ()
{
  digitalWrite (12, HIGH); // Turn on the LEDdelay (1000);
  digitalWrite (12, LOW); //Turn of the LEDdelay
  (1000);
}
```

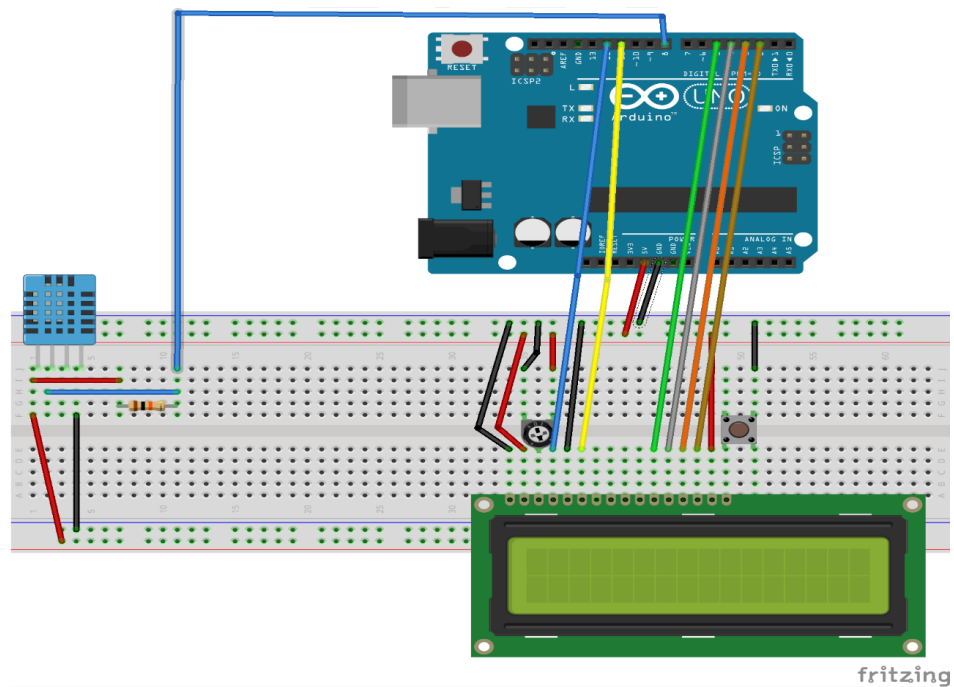
Results:

Aim: Interfacing DHT11 humidity sensor with Arduino.

Components Required:

S.No	Component	Quantity
1	Arduino uno	1Nos
2	Bread Board	1Nos
3	16 x 2 LCD display	1Nos
4	Male to male, male to female connectors	
5	Software Arduino IDE	1Nos
6	10 k ohm variable resistor pot	1Nos
7	DHT11 temperature and humidity sensor	1Nos
8	Usb communication cable of Arduino uno	1Nos

Connection Diagram:



Procedure:

1. The four data Pins D4 to D7 are connected to the four pins (2 to 5) of the arduino.
2. Rs (register select) and E (Enable) pins are connected to the pin12 and pin11 of the arduino.
3. V_{SS} pin of the LCD is connected to the ground while V_{DD} is connected to the power supply.
4. V_{EE} of LCD is connected to the potentiometer in order to vary the brightness of the LCD.
5. RW pin is connected to ground.
6. Download DHT11 sensor library from github and save it in the Arduino libraries and then include it in the code
7. Connect the dht11 wiring as shown in the circuit diagram

Program:

```
#include <LiquidCrystal.h>#include "DHT.h"
#define DHTPIN 8
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup()
{
    lcd.begin(16, 2);
    dht.begin();
    lcd.print("Temp:   Humidity:");
}

void loop()
{
    delay(500); lcd.setCursor(0, 1);
    float h = dht.readHumidity();

    float f = dht.readTemperature(true);if (isnan(h) ||
    isnan(f))
    {
        lcd.print("ERROR");return;
    }
    lcd.print(f); lcd.setCursor(7,1);
    lcd.print(h);
}
```

Results:

INTERFACING PIR SENSOR WITH ARDUINO

EXPT. NO :

DATE :

Aim: Intruder detection using PIR sensor using Arduino

Components Required:

S.No	Component	Quantity
1.	Arduino uno	1Nos
2.	LED	1Nos
3.	Bread Board	1Nos
4.	Male to male, male to female connectors	
5.	Software Arduino IDE	
6.	PIR sensor	1Nos
7.	Usb communication cable of Arduino uno	1Nos

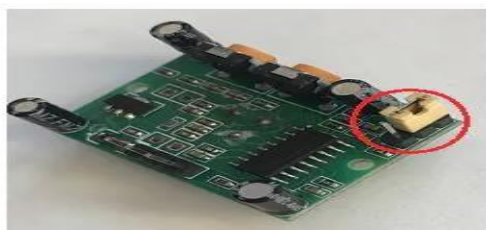
Working of PIR Sensor:

PIR sensor is a special type sensor which is usually used for security purposes. It detects the objects by reading the Infrared radiations emitted by the objects. Any object whose temperature is above absolute zero, emits radiation. This radiation is not visible to human eyes. The PIR sensor is designed to detect this Infrared radiation.

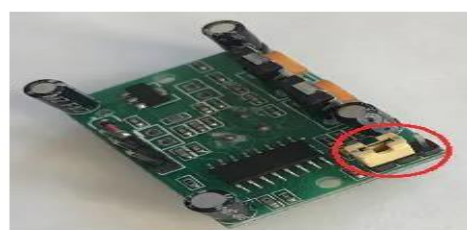


The PIR sensor has two modes. You can switch between these modes by interchanging the jumper behind the PIR sensor as shown in the images below.

1. Single Trigger mode

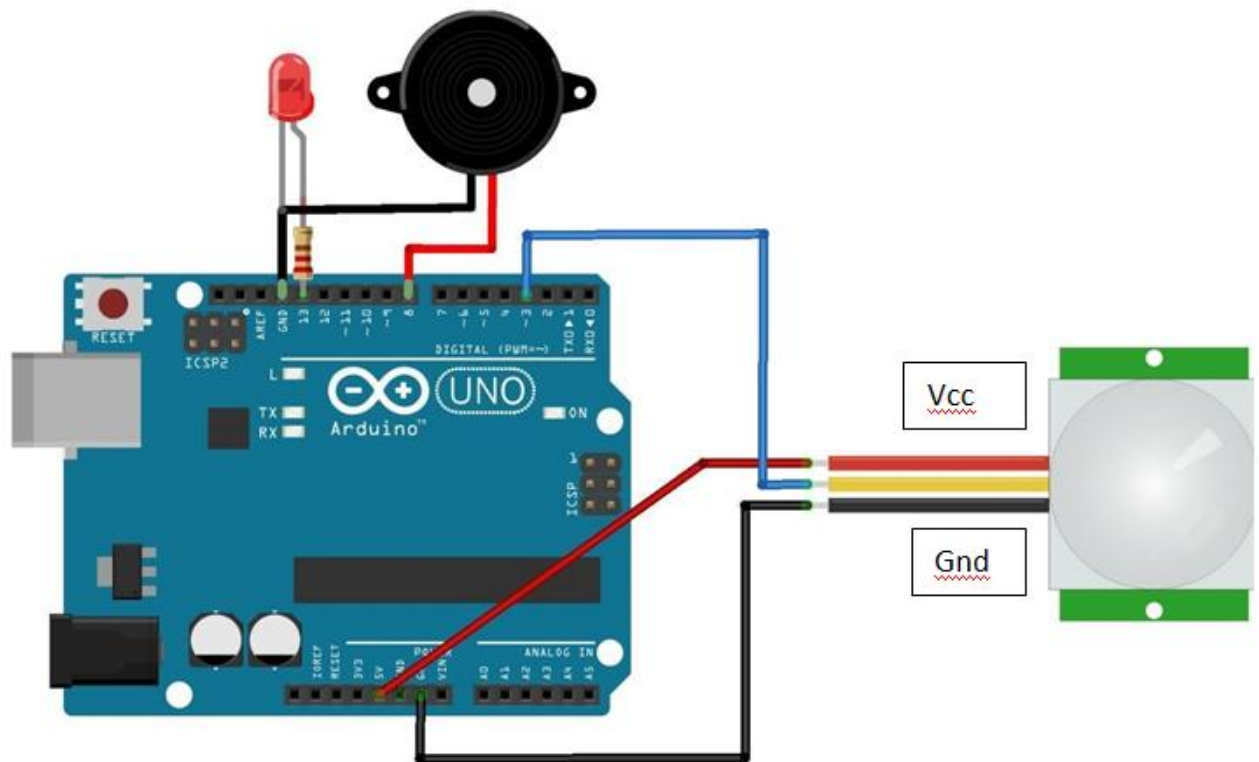


2. Repeatable Trigger mode



Procedure:

1. Interfacing PIR sensor with Arduino



1. PIR to Arduino

- Connect the Vcc of PIR to 5V on Arduino
- Connect the GND of PIR to GND on Arduino
- Connect the OUTPUT pin of PIR to Digital pin D3 on Arduino

2. Buzzer to Arduino

- Connect one pin of buzzer to digital pin D8 on Arduino
- Connect other pin of buzzer to GND on Arduino

3. LED to Arduino

- Connect the LED positive to Digital pin D13 on Arduino through a resistor.
- Connect the LED negative to GND on Arduino.

Program:

```
int Buzz= 8; // Define Buzzer pin
int LED= 13; // Define LED pin
int PIR= 3; // Define PIR pin

int val= 0; // Initializing the value as zero at the beginning

void setup()
{
  pinMode(Buzz, OUTPUT);
  pinMode(LED, OUTPUT);
  pinMode(PIR, INPUT);
  Serial.begin(9600);
}

void loop()
{
  val = digitalRead(PIR); // The value read from PIR pin
  if(val == HIGH)
  {
    digitalWrite(LED, HIGH); // Turn LED ON
    digitalWrite(Buzz, HIGH); // Turn Buzzer ON
    Serial.println("Movement Detected"); // Print this text in Serial Monitor
  }
  else
  {
    digitalWrite(LED, LOW);
    digitalWrite(Buzz, LOW);
    Serial.println("Movement not Detected");
  }
}
```

Results:

AIM: Simulate a traffic light using an Arduino and LED's.

Components Required:

S.No	Component	Quantity
1.	Arduino uno	1Nos
2.	LED (Red, Yellow & Green)	3Nos
3.	Bread Board	1Nos
4.	Male to male, male to female connectors	
5.	Software Arduino IDE	
6.	USB communication cable of Arduino uno	1Nos
7.	100ohm resistor	3Nos

Procedure:

Step 1: Supply power to the breadboard.

Step 2: Adding the LEDs.

Step 3: Completing the circuit.

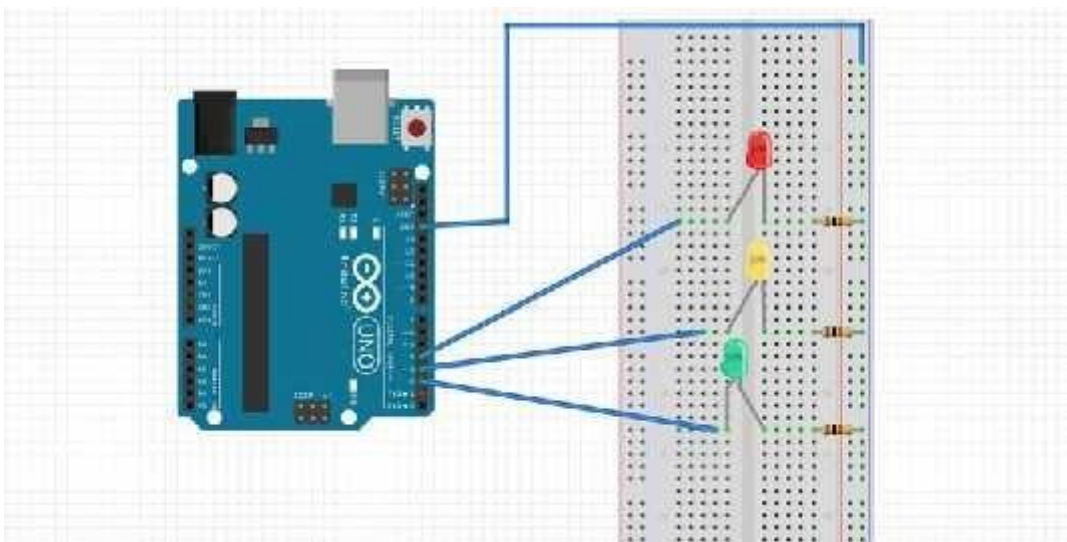
Step 4: Take another jumper wire, put it on the same row that you have an LED on.

This is where the wires will go:

Green LED: Port 2, Digital PWM section

Yellow LED, Port 3, Digital PWM section

Red LED, Port 4, Digital PWM section



Connection Diagram

Program to Interface Traffic Lights:

```
// variables
int GREEN = 2; int YELLOW = 3; int RED = 4;
int DELAY_GREEN = 5000; int DELAY_YELLOW = 2000; int DELAY_RED = 5000;
// basic functions
void setup()
{
  pinMode(GREEN, OUTPUT); pinMode(YELLOW, OUTPUT); pinMode(RED, OUTPUT);
}
void loop()
{
  green_light();
  delay(DELAY_GREEN);
  yellow_light();
  delay(DELAY_YELLOW);
  red_light();
  delay(DELAY_RED);
}
void green_light()
{
  digitalWrite(GREEN, HIGH); digitalWrite(YELLOW, LOW); digitalWrite(RED, LOW);
}
void yellow_light()
{
  digitalWrite(GREEN, LOW);
  digitalWrite(YELLOW, HIGH);
  digitalWrite(RED, LOW);
}
void red_light()
{
  digitalWrite(GREEN, LOW); digitalWrite(YELLOW, LOW); digitalWrite(RED, HIGH);
}
```

Results:

AIM: use one water flow sensor with an Arduino board.

Components Required:

1. Arduino uno
2. Water flow sensor
3. breadboard cables

Water flow Sensor:

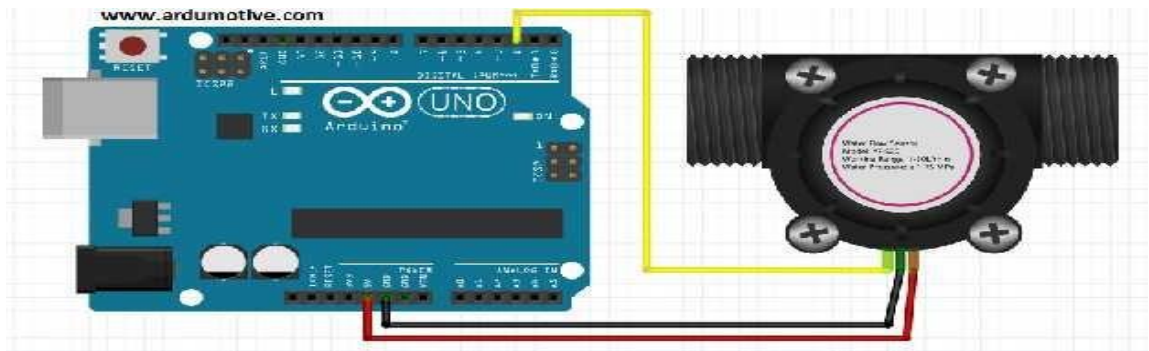
The water flow sensor consists of a plastic valve body, a water rotor and a hall-effect sensor. When the water flows through the rotor, rotor rolls and the speed of it changes with a different rate of flow. The hall-effect sensor outputs the corresponding pulse signal.

This type of sensor can be found on different diameters, water pressure (MPa) and flow rate (L/m) ranges. Make sure to select one that will cover your needs. The sensor that I have it has 20mm diameter, <1.75Mpa water pressure and ~30 L/m flow rate range.

In this, the serial monitor for printing the water flow rate in liters per hour and the total of liters flowed since starting. Press the connect button below to start the serial communication. Connect this sensor with your water tap, or just blow on it.



Circuit Diagram:



Program to Interface:

```
byte statusLed      = 13; byte sensorInterrupt = 0; byte sensorPin = 2;
float calibrationFactor = 4.5; volatile byte pulseCount; float flowRate;
unsigned int flowMilliLitres; unsigned long totalMilliLitres; unsigned long oldTime;

void setup()
{
    Serial.begin(9600);
    pinMode(statusLed, OUTPUT);
    digitalWrite(statusLed, HIGH); pinMode(sensorPin, INPUT); digitalWrite(sensorPin, HIGH); pulseCount = 0;
    flowRate = 0.0; flowMilliLitres = 0; totalMilliLitres = 0;
    oldTime = 0; attachInterrupt(sensorInterrupt, pulseCounter, FALLING);
}

void loop()
{
    if((millis() - oldTime) > 1000)
    {
        detachInterrupt(sensorInterrupt);
        flowRate = ((1000.0 / (millis() - oldTime)) * pulseCount) / calibrationFactor; oldTime = millis();
        flowMilliLitres = (flowRate / 60) * 1000; totalMilliLitres += flowMilliLitres; unsigned int frac;
        Serial.print("Flow rate: "); Serial.print(int(flowRate)); Serial.print("L/min"); Serial.print("\t");
        Serial.print("Output Liquid Quantity: "); Serial.print(totalMilliLitres); Serial.println("mL");
        Serial.print("\t"); // Print tab space
        Serial.print(totalMilliLitres/1000); Serial.print("L"); pulseCount = 0;
        attachInterrupt(sensorInterrupt, pulseCounter, FALLING);
    }
}

void pulseCounter()
{
    pulseCount++;
}
```

Result:

AIM: Measure distance using Ultrasonic sensor with an Arduino board.

Components Required:

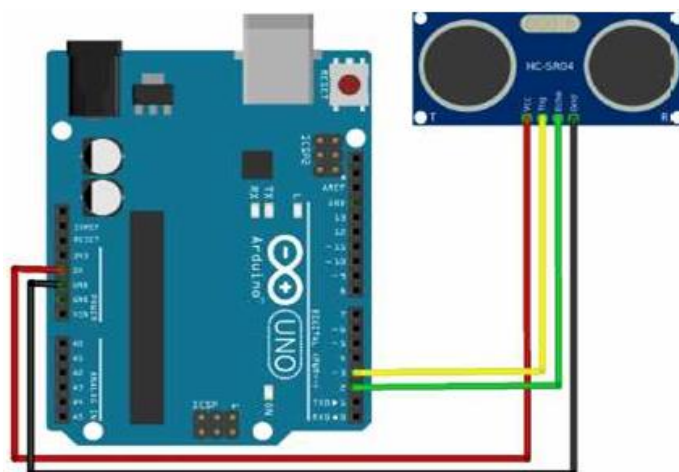
1. Arduino uno
2. Ultrasonic Sensor HC-SR04
3. Male to female Jumper Wires
4. breadboard

Ultrasonic Sensor HC-SR04:

Ultrasonic Sensor HC-SR04 is a sensor that can measure distance. It emits an ultrasound at 40 000 Hz (40kHz) which travels through the air and if there is an object or obstacle on its path It will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance.



The configuration pin of HC-SR04 is VCC (1), TRIG (2), ECHO (3), and GND (4). The supply voltage of VCC is +5V and you can attach TRIG and ECHO pin to any Digital I/O in your Arduino Board.



Connection Diagram:

The connection of Arduino and Ultrasonic Sensor HC-SR04

1. Connect the circuit as shown in the picture.
2. Open Arduino IDE Software and write down the code.
3. Choose Arduino board (in this case Arduino Uno), byselecting Tools > Board > Arduino/Geniuno Uno
4. Choose COM Port (usually it appears only one existing port),
Tools > Port > COM.. (If there are more than one ports, try it one by one
5. Sketch > Upload
6. To display the measurement data in Serial Monitor

Program to Interface:

```
// defines pins numbersconst
int trigPin = 9; const int
echoPin = 10;
// defines variableslong
duration;
int distance; void
setup() {
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
Serial.begin(9600); // Starts the serial communication
}
void loop() {
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the sound wave travel time in microsecondsduration =
pulseIn(echoPin, HIGH);
// Calculating the distance distance=
duration*0.034/2;
// Prints the distance on the Serial Monitor
Serial.print("Distance: "); Serial.println(distance);
}
```

Results:

INTERNET OF THINGS (IOT)

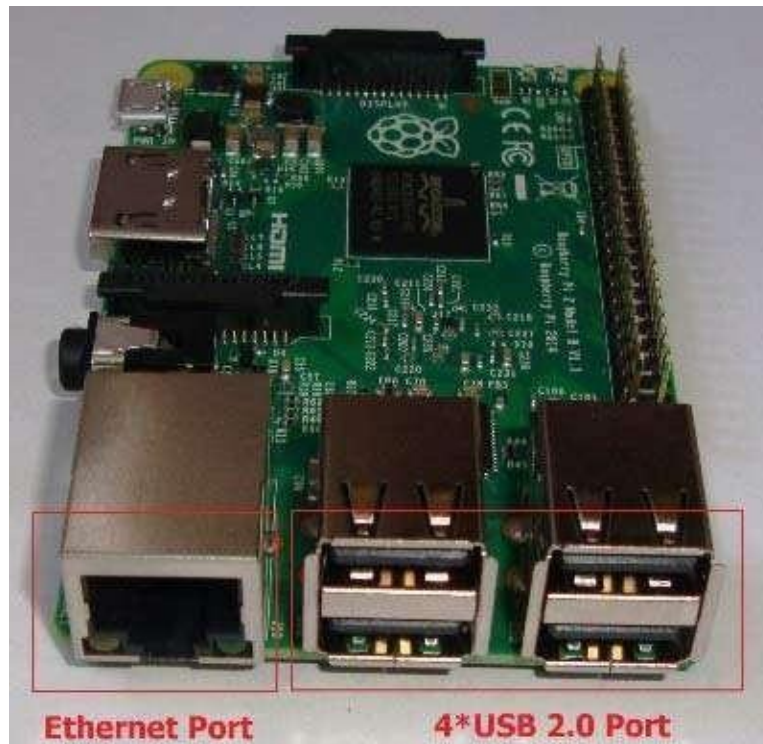
Using

Raspberry Pi

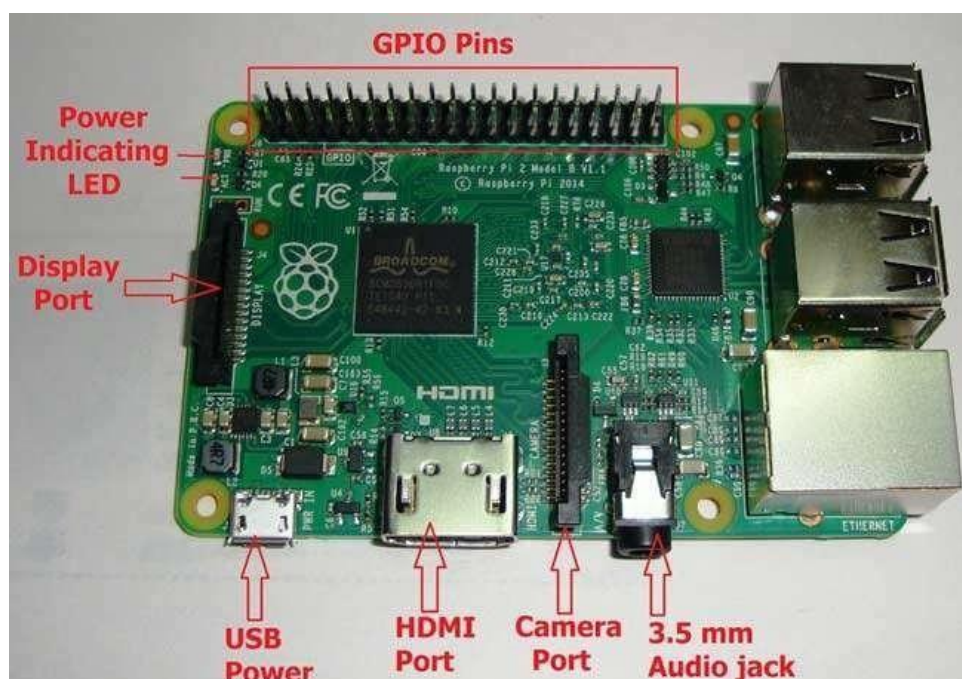


Getting started with Raspberry Pi

Raspberry Pi is an ARM cortex based board designed for Electronic Engineers and Hobbyists. It's a single board computer working on low power. With the processing speed and memory, Raspberry Pi can be used for performing different functions at a time, like a normal PC, and hence it is called Mini Computer in your palm. Because it has an ARMv7 processor, it can run the full range of ARM GNU/Linux distributions as well as Microsoft Windows 10, we will discuss about that later. ARM architecture is very influential in modern electronics. We are using the ARM architecture based processors and controllers everywhere. For example we are using ARM CORTEX processors in our mobiles, iPods and computers etc. Pi is an amazing tool for realizing „Internet of Things“. In this session we will discuss the hardware and software requirements for Pi and setting up the Operating System for the first run of PI. There are different types of Raspberry Pi boards in the market now, with **Raspberry Pi 2 Model B** being the most popular. **Raspberry Pi 3 Model B** has also been launched; it is almost similar to RPi 2, with some advance feature like on board Wi-Fi and Bluetooth connectivity, more powerful CPU etc. We will discuss few characteristics of “Raspberry Pi 2 B” now. Raspbeery Pi has **four USB 2.0 ports**. These ports can be connected to any USB devices, like mouse and keyboard. With the first start itself, we need mouse and keyboard, wewill discuss it later. The four USB ports are shown the figure. Raspbeery Pi 2 has one **Ethernet port**. This port is for internet connectivity to the RASPBEERY PI 2. This Ethernet port can also be used to transfer data between PI2 and your PC. It has a **3.5mm jack** port for connecting headphones, in case of playing music from Pi. PI has a **single HDMI port** for connecting a LCD/LED screen. The graphics provided by the chip is fairly good



There is a **micro USB port** on the board; we provide power for the complete board through this port. If there are any fluctuations in voltage provided at this port, the board will not work properly. Instead of connecting an LCD screen we can connect a 3inch to 7inch touch display. We have inbuilt **port for connecting a touch display**. We have a similar **port for connecting a camera** to the module; the camera module can be connected to PI without any additional attachments.



There are **GPIO** (General Purpose Input Output) pins and a couple power ground terminals. We can program there GPIO pins for any use. Few of these pins also perform special functions, we will discuss them later.

Hardware Requirements:

1. Power supply - As said earlier we will power the Raspberry Pi board by **Micro USB port** present on the board. Under normal operations the PI board needs a 5V, 1000mA (or 1A) power source. The voltage and current requirements are important here. Any power source higher than 5V will damage the board permanently and for voltages lower than 4.8V, the board will not function. Here I am using a **5V, 1000mA mobile phone charger** for powering up my PI. Remember the minimum current rating for normal operation of PI board.



For connecting the micro USB power, you need a good quality cable. If you do not power the board from a good **USB cable**, no matter what the power source, you will always have power shortage on the board. You need a good quality USB cable as shown in figure.

For higher operations of PI, you need a power source which could deliver at least 2000mA or 2A. So if you don't have a power source of such kind, don't drive the PI by lowerrated power source, its better get a new one.

But if you have **two adapters** which can provide each, you can connect one adapter output to the micro USB, and the second one to the USB 2.0 port, they both can share the load. Here I have a 0.7A or 700mA adapter which I connect to one of 4 USB ports on the chip.



1. You need a **LCD or LED screen**, you can use your old PC screen as a Raspberry Pi screen. After choosing your screen, you have to look whether the screen supports HDMI inputs or not. If your screen has a HDMI port then you just need to get a male to male **HDMI cable** as shown in figure.



If your screen does not support HDMI like mine, then your screen must have **VGA support** as shown in figure. You need a **HDMI to VGA converter**; you can buy this at any electronic store. This device converts HDMI from PI to VGA output. So we can interface a **VGA monitor** to a PI. The device is shown in figure.



2. You need a **Mouse and Keyboard**, make sure they are USB driven type or you won't be able to connect it to PI, since PI only has USB ports.

3. You need a **Micro SD card** (Memory card) and a **SD Card Reader** (or Adapter) to connect SD card to PC (or laptop). The SD card must be of 8GB or higher. If not, you won't be able to install the OS (Operating System) on to the PI easily. And also the Class of SD card should be equal or higher than 4, for better speed. "Speed Class" represents the writing speed like class 10 means 10 MB/second.



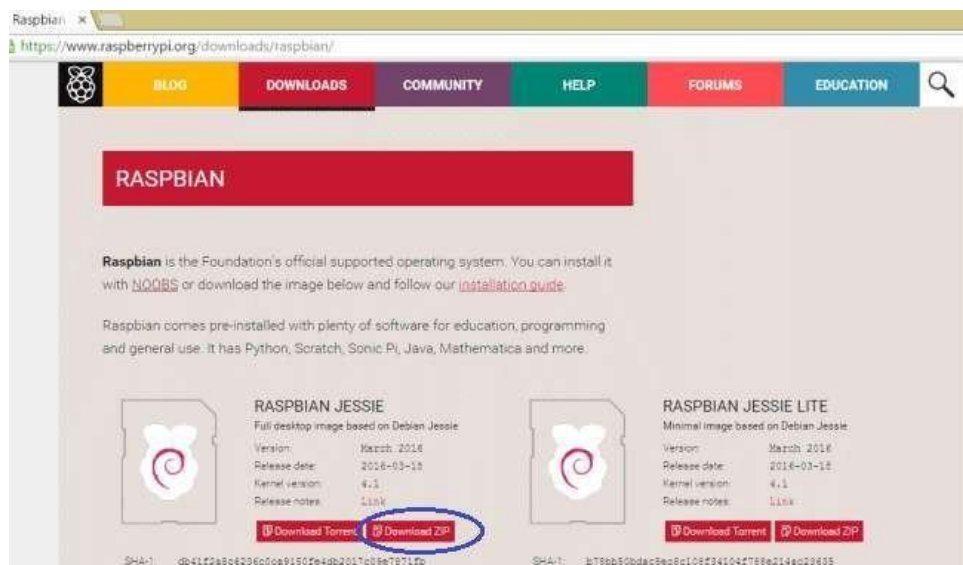
Now we have all the basic Hardware's, need to **Getting Started with Raspberry Pi**, and we will now discuss the Software Requirement.

Software Requirements:

First we need the OS (Operating System) for the PI, which can be downloaded from Downloads section of Raspberry Pi website:

<https://www.raspberrypi.org/downloads/>

It will show you all the supporting OS for the **RASPBERRY PI 2**. You can download and install any OS on Pi which is listed there. We are going to download official supported Operating System for Raspberry Pi, which is “**Raspbian**”. Click on “**RASPBIAN**”, and download the Full desktop image of **Raspbian Jessie**. Extract the Raspbian image from the Zip file, using any Zip file extractor like Winrar or Winzip.



We also need a **Image writer software** for installing the OS on to the Micro SD card. We have used “**win32diskimager**” to write the image on Micro SD card, which can be downloaded from below link:

<https://sourceforge.net/projects/win32diskimager/>

Once the download completes, install the software, you will see an icon on the Desktop Screen after installation.

Get started with Raspberry Pi: Steps

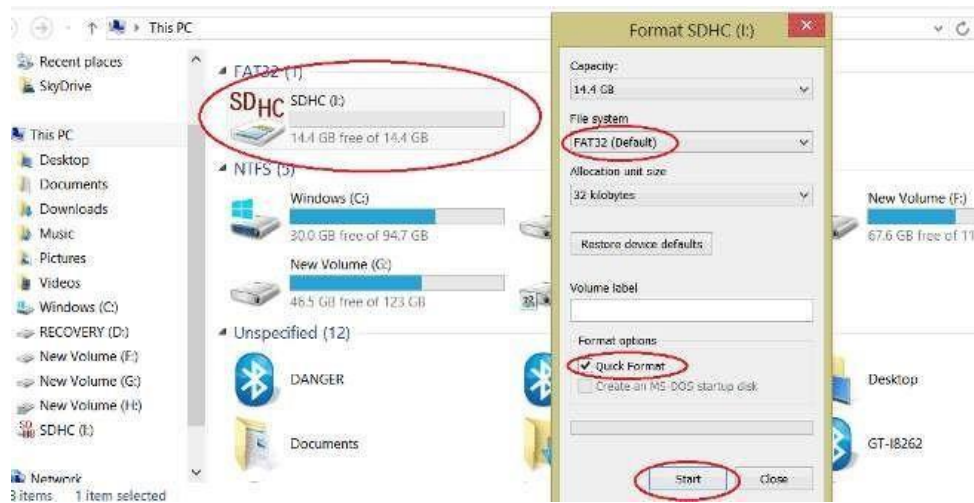
Now, we have all the **software and hard ware required to get started with the RASPBERRY PI 2**.

To install OS on to a SD card follow **below steps**:

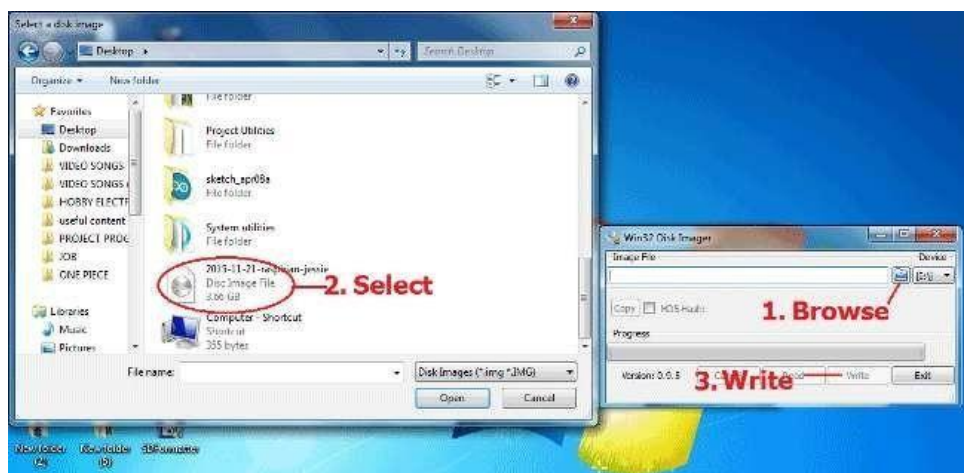
1. **UNZIP** the „**Raspbian Jessie**“ (OS ZIP file we downloaded from raspberry website) on to the desktop; you will see an Image icon upon extraction on the screen as shown below. Make sure you have at least 5 GB free disk space on „C“ drive of your PC. The extraction file size would be greater than 3GB.
2. Insert the SD card into the USB card reader or Card Adapter. Plug the card reader to the PC.

You must see the card on the screen as shown below.

3. Format the card drive by Right click on it and select **Format**. Select File system as „FAT32“ and tick on „Quick Format“. Finally click on „Start“ button to Format the drive.

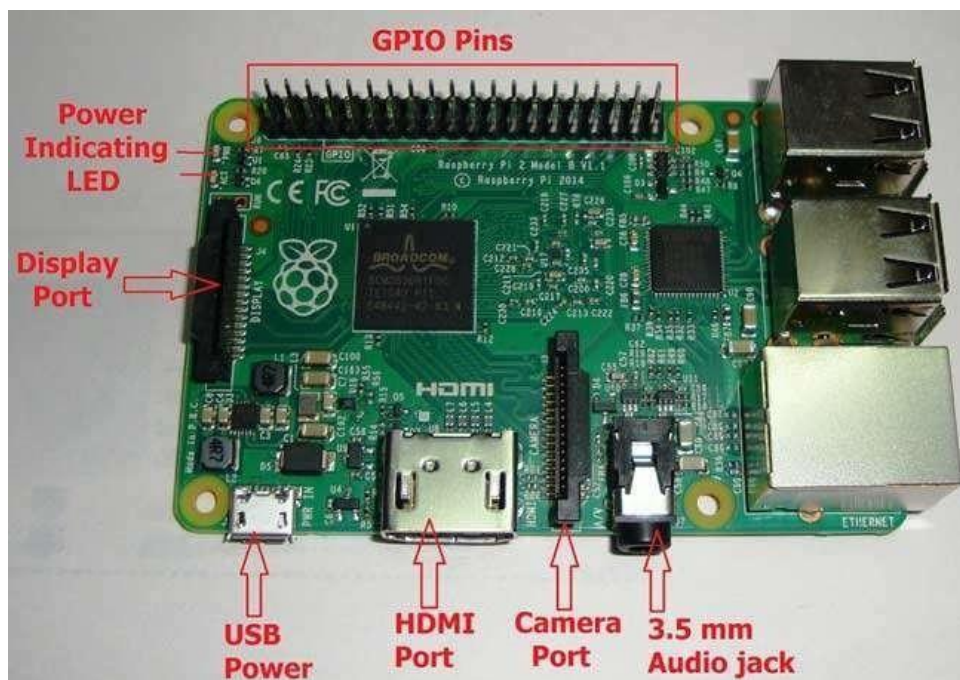


4. After formatting, run the “win32diskimager” application, which we have downloaded as explained previously.
5. Choose the SD card drive, browse for the Raspbian OS image file (which is extracted on the screen) and Click on ‘**WRITE**’ icon, to start writing the extracted OS file on to the SD card. This is shown in below figure.



6. After completion of writing, safely remove the SD card from the reader.

Now we have **SD card with Raspbian OS installed** on it and having all the equipment required to Get Started with Raspberry PI 2. In the next session we will have the first look at the “PI” OS and we will talk about configuring the BIOS of Raspberry Pi.



In previous session, we have learned about Raspberry Pi board, its ports, its hardware and software requirements and learned to install OS on SD card for PI. Once the OS (**Raspbian Jessie**) installed SD card is inserted into the Raspberry Pi with the screen, keyboard and mouse connected, we are ready to **boot the Jessie first time**. For this particular OS you don't need Ethernet connection. Once the power is started ON, you will see the power RED LED glowing. The BLUE LED will start blinking at this stage, this means the OS (Operating System) is loading and the PI is checking all the drivers. By this time you will see data on the screen as shown below,

```

[ 3.286397] (<001cf490>) (ext4_lookup) from (<0014c018>) (lookup_real+0x30/0x5c)
[ 3.297283] (<0014c018>) (lookup_real) from (<0014cbb0>) (lookup_hash+0x44/0x4c)
[ 3.308355] (<0014cbb0>) (lookup_hash) from (<0014ec38>) (lookup_slow+0x48/0x64)
[ 3.319425] (<0014ec38>) (lookup_slow) from (<0014fb0c>) (path_lookupat+0x6e8/0x730)
[ 3.330670] (<0014fb0c>) (path_lookupat) from (<0014ff98>) (filename_lookup.isra.46+0x30/0x70)
[ 3.342818] (<0014ff98>) (filename_lookup.isra.46) from (<00152130>) (user_path_at_empty+0x64/0x8c)
[ 3.355400] (<00152130>) (user_path_at_empty) from (<0015217c>) (user_path_at+0x24/0x2c)
[ 3.367063] (<0015217c>) (user_path_at) from (<00141f18>) (SyS_faccessat+0xa0/0x1d8)
[ 3.378370] (<00141f18>) (SyS_faccessat) from (<00142070>) (SyS_access+0x28/0x2c)
[ 3.389433] (<00142070>) (SyS_access) from (<0000ebc0>) (ret_fast_syscall+0x0/0x48)
[ 3.400661] Code: e7934004 e3540000 0a00004c e5963014 (e794e003)
[ 3.410367] ---[ end trace dc0385eb0d5102c5 ]---
[ 3.413518] usb 1-1: New USB device found, idVendor=0424, idProduct=9514
[ 3.413525] usb 1-1: New USB device strings: Mfr=0, Product=0, SerialNumber=0
[ 3.414105] hub 1-1:1.0: USB hub found
[ 3.414178] hub 1-1:1.0: 5 ports detected
[ 3.454547] Kernel panic - not syncing: Attempted to kill init! exitcode=0x0000000b
[ 3.454547]
[ 3.470602] CPU0: stopping
[ 3.476942] CPU: 0 PID: 0 Comm: swapper/0 Tainted: G      D      3.18.7-07+ #755
[ 3.487908] (<00016d14>) (unwind_backtrace) from (<00012c40>) (show_stack+0x20/0x24)
[ 3.499161] (<00012c40>) (show_stack) from (<0052efc8>) (dump_stack+0x90/0xd8)
[ 3.509881] (<0052efc8>) (dump_stack) from (<0001509c>) (handle_IP1+0x234/0x268)
[ 3.520759] (<0001509c>) (handle_IP1) from (<00000618>) (do_IP1+0x18/0x1c)
[ 3.531122] (<00000618>) (do_IP1) from (<005349b4>) (irq_svc+0x34/0x14c)
[ 3.541471] Exception stack(0x807cbf08 to 0x807cbf50)
[ 3.550003] bf00: 007c9ccc 00000000 ffffffff 00000000 807ca020 807cbdd4

```

As told its just PI is loading all the drives. You have to wait until all the drivers are checked, in case of error, turn off and on to restart PI. If there is still trouble try installing the OS on to the SD card again by following the steps described in first session.

If everything goes successfully, you will be asked for authorization. This authorization is predefined, with Username “pi” and password “raspberrypi”,

```
USER: pi
< Press enter>
PASSWORD: raspberrypi
< Press enter>
```

Once you enter these details, you will be entered in CLI mode (Command Line) of RaspberryPi. For entering into the DESKTOP of PI you need to type,

```
startx
< Press enter>
```

BIOS settings:

Now you have entered into the screen of Raspberry Pi and you are ready to go. Before going for the programming,

1. You need to configure the BIOS settings of PI.
2. You need to configure the keyboard, you have chosen.

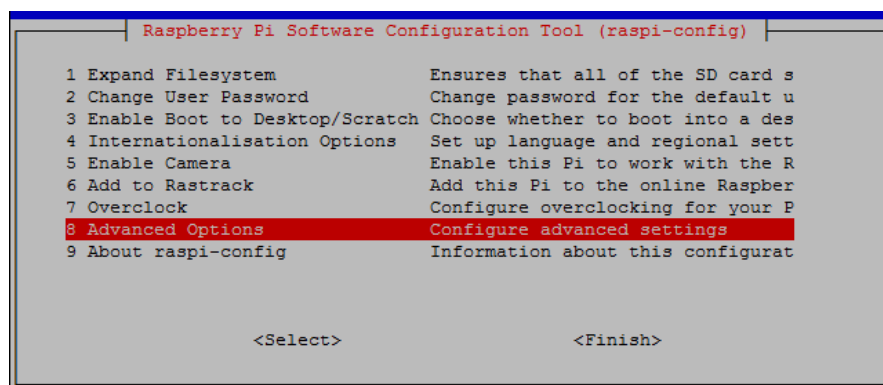
If you don’t do these two things first you get lot of errors, while programming and operating the PI.

For **configuring the BIOS of PI** first open the „LX TERMINAL” of PI and enter this

```
sudo raspi-config
< Press enter>
```

You will be entered into BIOS after this; you can watch the video below to see how it’s done.

The BIOS options are as,



We will discuss each of these options briefly below,

1. **Expand File System:** After first start, some memory of the SD will be misplaced and is not considered by PI. You need to select this option to get the files & order and to display the remaining memory of SD card. Once you choose this option, the PI will REBOOT to get everything in order. If you don't expand the file system, you will not be able to use remaining memory of SD card.
2. **Change Password:** This option changes the login password, its "pi" by default. Just leave it, if you are not doing any important work.
3. **Boot To Desktop:** On the first start you are entered in CLI mode, as discussed earlier. This option disables that, so that you can enter DESKTOP of Pi with every start.
4. **Internationalisation Options:** This option is for choosing language. Its ENGLISH by default, leave it you don't want to change the language and date. The DATE may be wrong, we will configure these settings later.
5. **Enable Camera:** If you have a camera module at hand, configure this option. It will turn on the CAM. If you don't have a camera, just leave it. Remember camera module draws power, so once you enable it the module will be drawing power continuously.
6. **Add To Rastack:** This option is for connecting your PI online. Just leave it.
7. **Over Clock:** This option over clocks PI, thus increasing your PI speed and also its power consumption. Over clocking might damage the board, if efficient cooling system is not provided. For basic programming, you need not over clock the PI, just leave it 900MHz.
8. **Advanced Options:** These options for BOARD devices (like AUDIO, I2C etc) configuration:
We have few options under this, but the important for now is, „AUDIO“. The PI can output AUDIO either from HEADPHONE jack (on the PI board) or from HDMI port. To find out the ports, check Exptt-1.

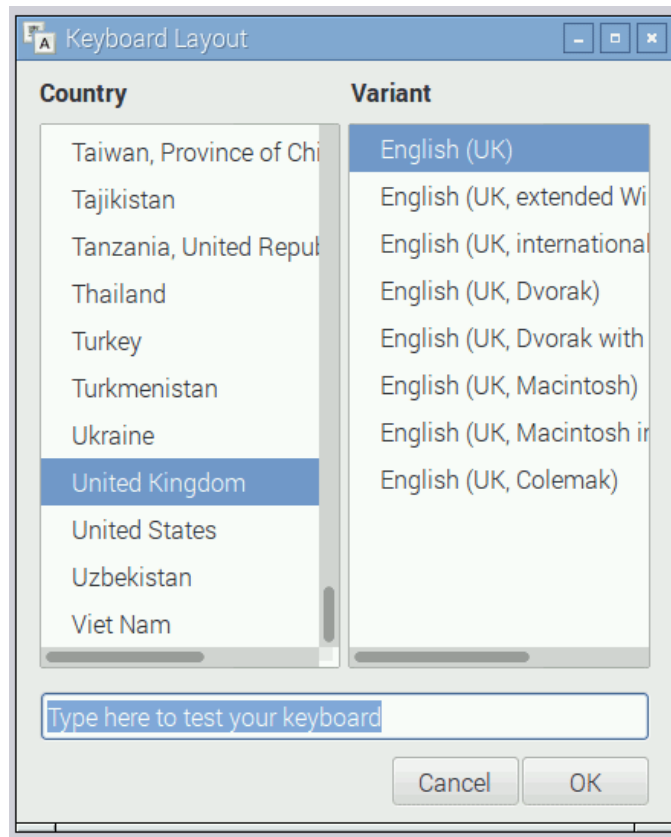
Choose the appropriate one based on your usage. If you won't configure this, you won't get AUDIO.

The remaining options are not important for now.

Keyboard Configuring:

We use different type of keyboard all around the world, most of the keys match for all keyboards. But few keys mismatch. This will cause really trouble while programming, as special keys play a crucial part while doing programming in **PYTHON and LINUX**. So we need to configure the Raspberry Pi to the keyboard we are using. I am from INDIA and almost everyone one here uses US (United States) keyboard models.

So I am configuring this on the „Keyboard Layout“ option,



Errors: Sometimes you will have white borders at the edges of the screen, you can select them by right mouse button and delete them by left mouse button

AIM: Blinking of LED Using Raspberry Pi.

Components Required:

Here we are using **Raspberry Pi 2 Model B with Raspbian Jessie OS**. All the basic Hardware and Software requirements are previously discussed, you can look it up in the **Raspberry Pi Introduction**, other than that we need:

1. 1 - Raspberry Pi device with a 5 V power supply
2. 1 - Bread Board
3. 1 - Male to Female jumper cable
4. 1 – LED

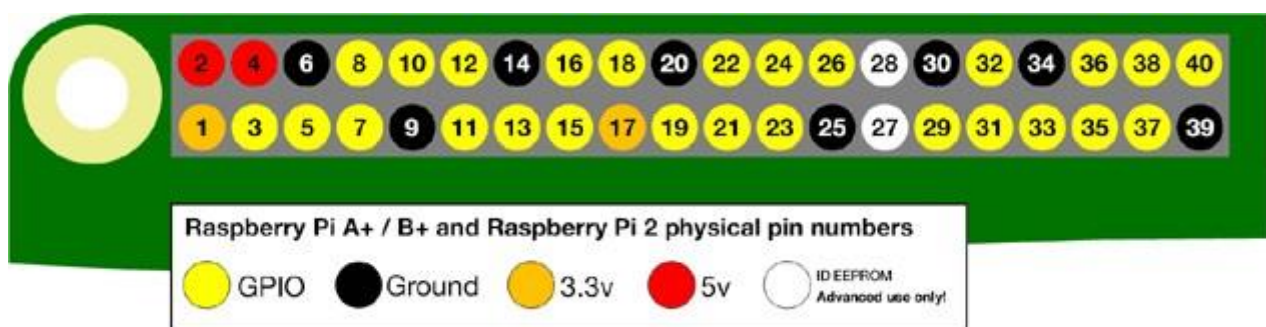
Raspberry Pi LED Blinking with Python Programming:

Raspberry Pi is an ARM architecture processor based board designed for electronic engineers and hobbyists. The PI is one of most trusted project development platforms out there now. With higher processor speed and 1 GB RAM, the PI can be used for many high profile projects like Image processing and **Internet of Things**.

For doing any of high profile projects, one need to understand the basic functions of PI. That is why we are here, we will be teaching all the basic functionalities of Raspberry Pi in these tutorials. In each tutorial series we will discuss one of functions of PI. By the end of tutorial series you will be able to do high profile projects by yourself.

In this tutorial of PI series, we will understand the **concept of writing and executing programs on PYTHON**. We will start with **Blink LED using Raspberry Pi**. Blinky is done by connecting an LED to one of GPIO pins of PI and turning it ON and OFF.

We will discuss a bit about **PI GPIO Pins** before going any further,



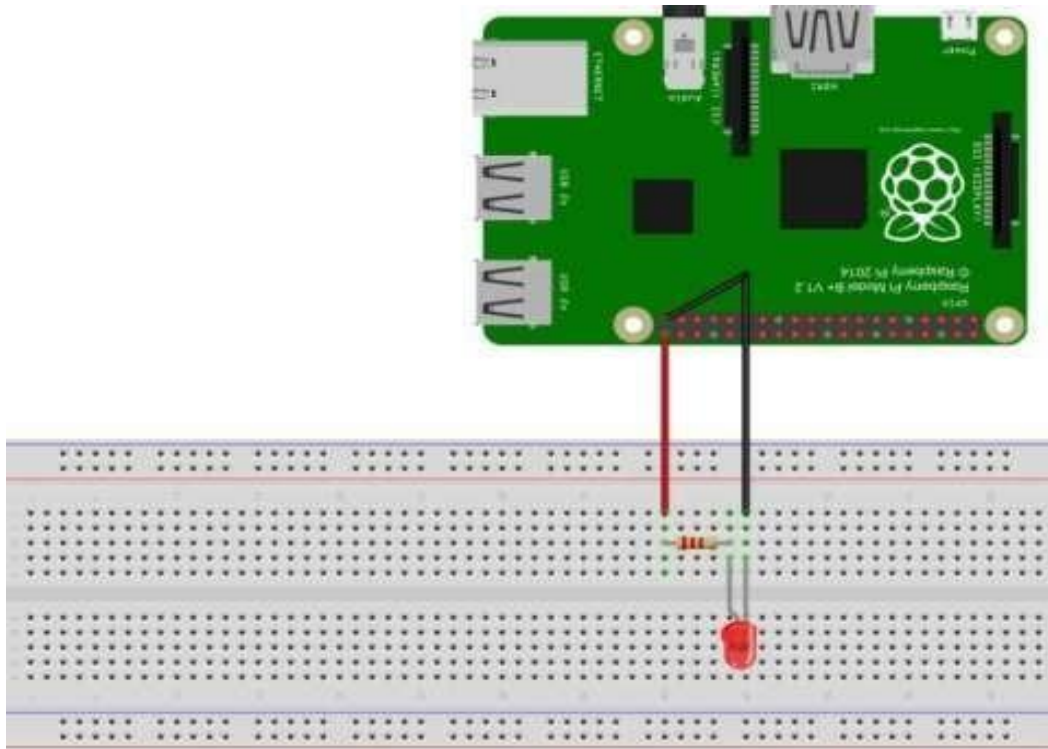
As shown in above figure, there are 40 output pins for the PI. But when you look at the second figure, you can see not all 40 pin out can be programmed to our use. These are only 26 GPIO pins which can be programmed. These pins go from **GPIO2 to GPIO27**.

These **26 GPIO pins can be programmed** as per need. Some of these pins also perform some special functions, we will discuss about that later. With special GPIO put aside, we have 17 GPIO remaining (Light green Girl).

Each of these 17 GPIO pins can deliver a maximum of **15mA current**. And the sum of currents from all GPIO cannot exceed 50mA. So we can draw a maximum of 3mA in average from each of these GPIO pins. So one should not tamper with these things unless you know what you are doing.



Circuit Explanation:

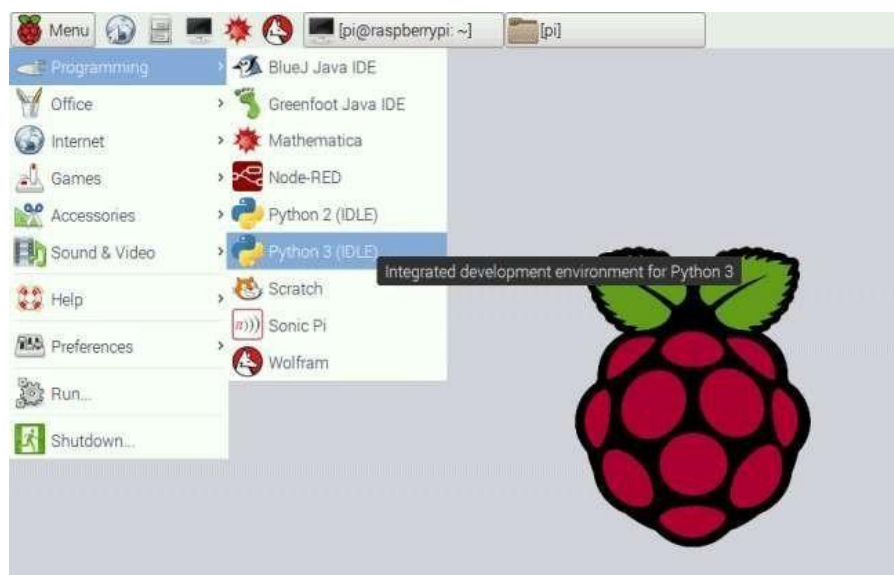


As shown in the circuit diagram we are going to connect an LED between PIN40 (GPIO21) and PIN39 (GROUND). As said earlier, we cannot draw more than 15mA from any one of these pins, so to limit the current we are connecting a 220 Ω or 1K Ω resistor in series with the LED.

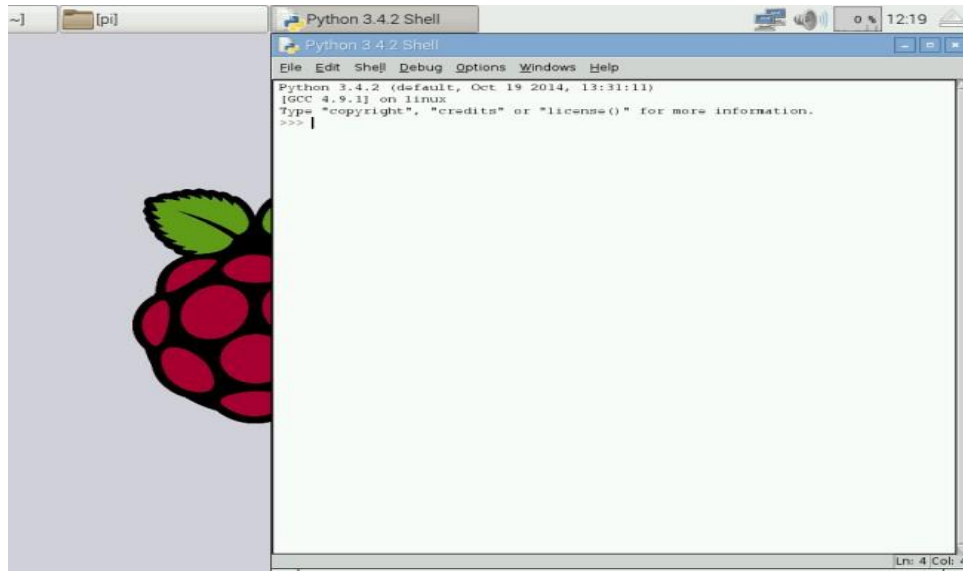
Working Explanation:

Since we have everything ready, turn ON your PI and go to the desktop.

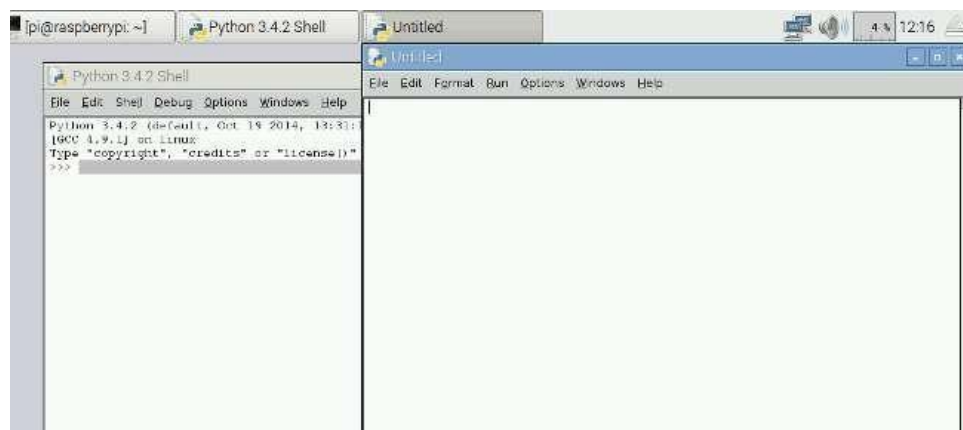
1. On the desktop, go the Start Menu and choose for the **PYTHON 3**, as shown in figure below.



1. After that, PYHON will run and you will see a window as shown in below figure.



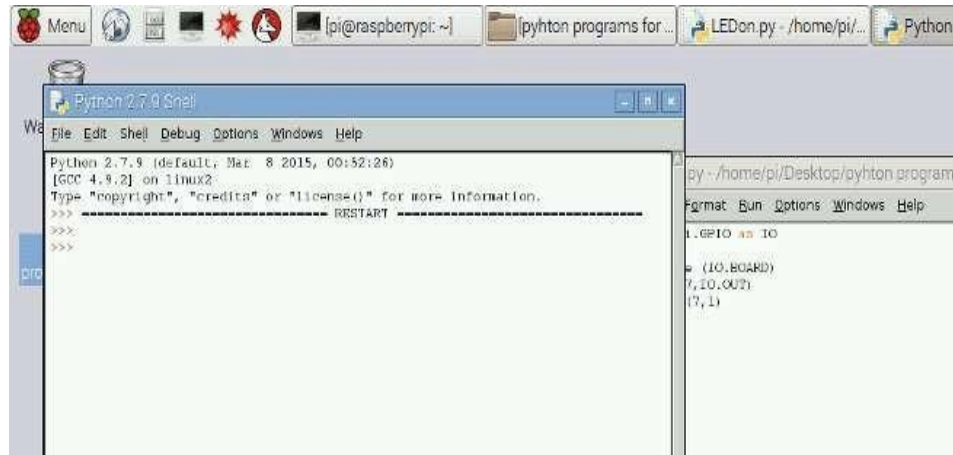
2. After that, click on *New File* in *File* Menu, You will see a new Window open,



3. Save this file as *blink* on the desktop,



4. After that write the program for *blink* as given below and execute the program by clicking on “RUN” on „DEBUG“ option.



If the program has no errors in it, you will see a “>>>”, which means the program is executed successfully. By this time you should see the LED blinking three times. If there were any errors in the program, the execution tells to correct it. Once the error is corrected execute the program again.

Connection Diagram:



Connection Procedure:

1. Firstly, connect the GPIO pin ‘8’ to the bread board through jumper cable.
2. Now take LED and put it on the bread board. Connect the plus terminal of LED to the Jumper cable coming from the GPIO Pin “8”.
3. Now take another jumper cable, connect it to the GPIO pin “**Ground**” and another terminal of this wire to minus terminal of the LED.
4. Now write the python program for blinking LED every 1 second.
5. Now run the program and observe output in Blinking of LED.

Python Code:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.OUT)
while True:
    GPIO.output(8, True)
    time.sleep(1)
    GPIO.output(8, False)
    time.sleep(1)
```

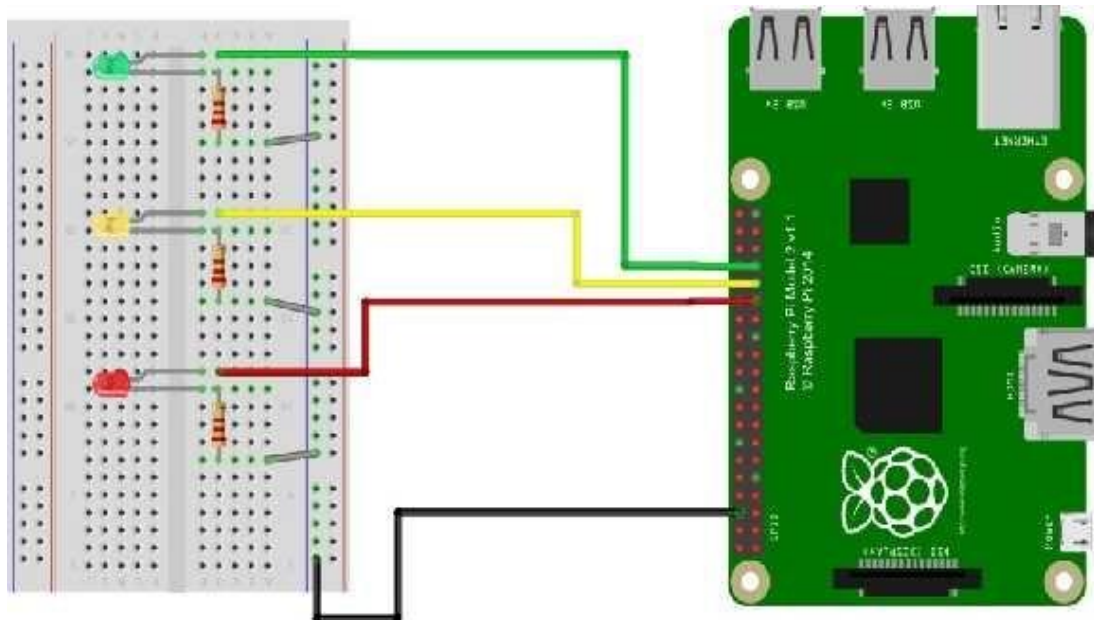
Result:

AIM: Simulate a traffic light using an Raspberry Pi and LED's.

Components Required:

S.No	Component	Quantity
1.	Raspberry Pi	1Nos
2.	LED (Red, Yellow & Green)	3Nos
3.	Bread Board	1Nos
4.	Male to female connectors	4Nos
5.	100ohm resistor	3Nos

Connection Diagram:



Procedure:

1. To get started, you'll need to place all the components on the breadboard and connect them to the appropriate GPIO pins on the Raspberry Pi.
2. An LED requires 1 ground pin and 1 GPIO pin, with a current limiting resistor.
3. Each component requires its own individual GPIO pin, but components can share a ground pin.
4. Place the components on the breadboard and connect them to the Raspberry Pi GPIO pins, according to the following diagram:
5. Note that the row along the long side of the breadboard is connected to a ground pin on the Raspberry Pi, so all the components in that row (which is used as a ground rail) are hence connected to ground.

Python code:

```
from gpiozero import LED
from time import sleep
red=LED(22)
blue=LED(27)
green=LED(17
)
while True:
    red. on()
    sleep (1)
    blue.
    on()
    sleep (1)
    green.
    on()sleep
    (1)
    red.off()
    sleep (1)
    blue.
    off()
    sleep (1)
    green.
    off()
    sleep(1)
```

Results:

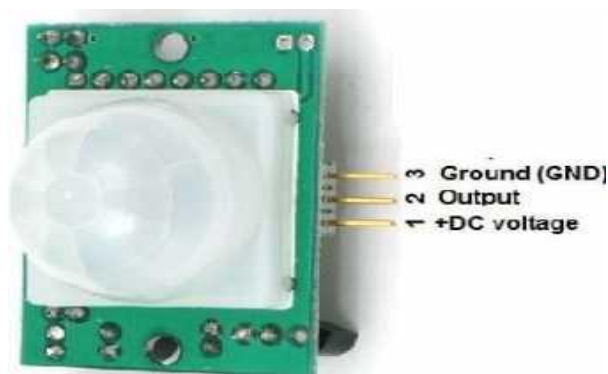
AIM: To Interface a PIR Motion Sensor With Raspberry Pi GPIO.

Components Required:

S.No	Component	Quantity
1.	Raspberry Pi	1Nos
2.	PIR Sensor	1Nos
3.	Bread Board	1Nos
4.	Male to female and Male to male connectors	4Nos
5.	LED	1Nos

Connection Diagram:

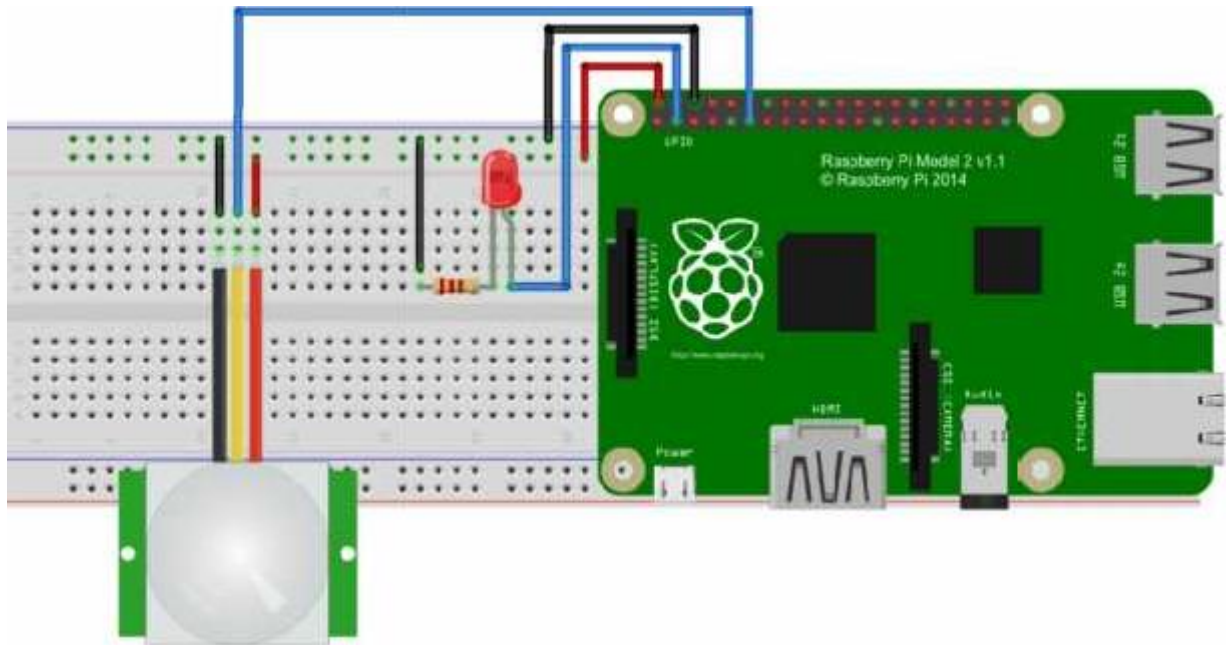
Here, we are using a PIR motion sensor. PIR stands for passive infrared. This motion sensor consists of a fresnel lens, an infrared detector, and supporting detection circuitry. The lens on the sensor focuses any infrared radiation present around it toward the infrared detector. Our bodies generate infrared heat, and as a result, this heat is picked up by the motion sensor. The sensor outputs a 5V signal for a period of one minute as soon as it detects the presence of a person. It offers a tentative range of detection of about 6–7 meters and is highly sensitive. When the PIR motion sensor detects a person, it outputs a 5V signal to the Raspberry Pi through its GPIO and define what the Raspberry Pi should do as it detects an intruder through the Python coding. Here we are just printing "Intruder detected".



Procedure:

1. Connect the PIR sensor's pin labelled VCC to the 5V pin on the Raspberry Pi. This provides power to the PIR sensor.
2. Connect the one labelled GND to a ground pin on the Pi (also labelled GND). This completes the circuit.

3. Connect the one labelled OUT to any numbered GPIO pin on the Pi.



Python Code:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.IN)
GPIO.setup(3, GPIO.OUT) #Read output from
while True: #LED output pin
    i=GPIO.input (11)
    if i==0:
        print ("No intruders", i) #When output from motion sensor
        GPIO.output(3, 0)
        time.sleep(0.1) #Turn OFF LED
    elif i==1:
        print ("Intruder detected", i) #When output from motion .
        GPIO.output (3, 1) #Turn ON LED
        time.sleep (0.1)
```

Results:

AIM: Find distance using Ultra Sonic Sensor with Raspberry Pi GPIO.

Components Required:

S.No	Component	Quantity
1.	Raspberry Pi	1Nos
2.	Ultra Sonic Sensor	1Nos
3.	Bread Board	1Nos
4.	Male to female and Male to male connectors	4Nos

Ultrasonic Sensor HC-SR04:

Ultrasonic Sensor HC-SR04 is a sensor that can measure distance. It emits an ultrasound at 40 000 Hz (40 kHz) which travels through the air and if there is an object or obstacle on its path It will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance.



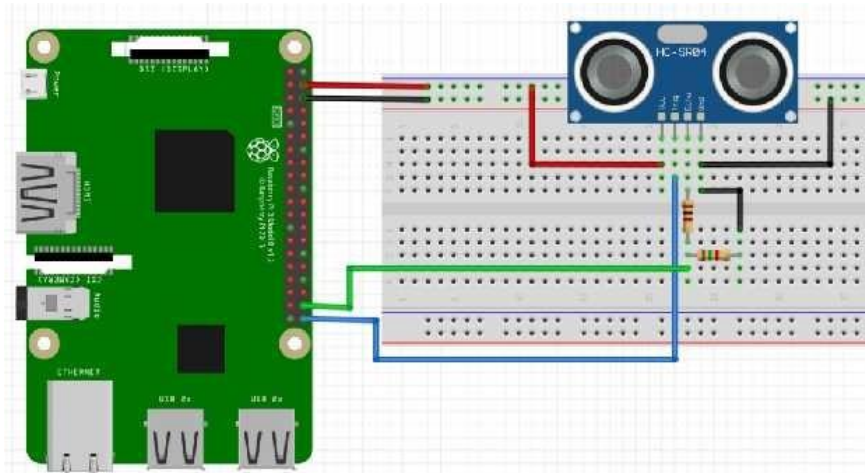
The configuration pin of HC-SR04 is VCC (1), TRIG (2), ECHO (3), and GND (4). The **supply voltage** of VCC is **+5V** and you can attach TRIG and ECHO pin to any Digital I/O in your Arduino Board.

Procedure:

There are four pins on the ultrasound module that are connected to the Raspberry:

- VCC to Pin 2 (VCC)
- GND to Pin 6 (GND)
- TRIG to Pin 12 (GPIO18)
- Connect the 330Ω resistor to ECHO. On its end you connect it to Pin 18 (GPIO24) and through a 470Ω resistor you connect it also to Pin6 (GND).

We do this because the GPIO pins only tolerate maximal 3.3V. The connection to GND is to have a obvious signal on GPIO24. If no pulse is sent, the signal is 0 (through the connection with GND), else it is 1. If there would be no connection to GND, the input would be undefined if no signal is sent (randomly 0 or 1), so ambiguous.



Python Code:

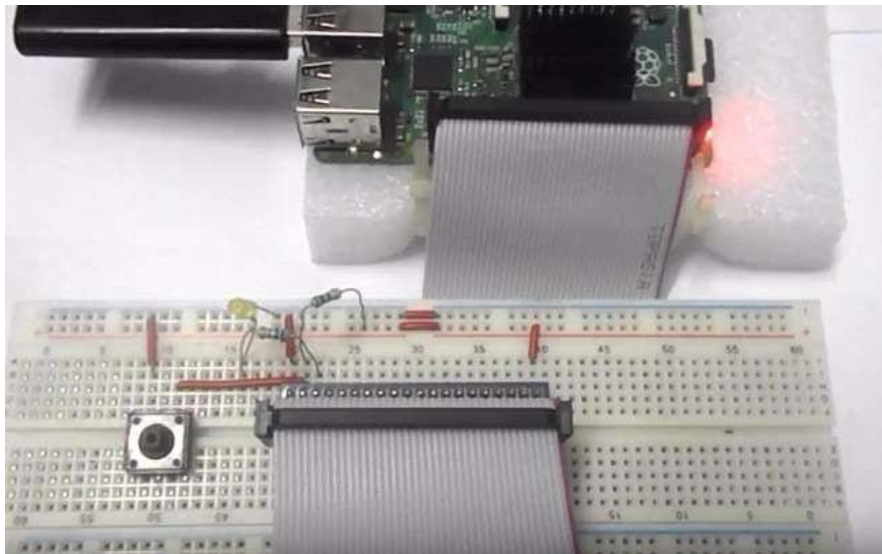
```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO_TRIGGER = 18
GPIO_ECHO = 24
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)

def distance():
    GPIO.output(GPIO_TRIGGER, True)
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)
    StartTime = time.time()
    StopTime = time.time()
    while GPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()
    while GPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()
    TimeElapsed = StopTime - StartTime
    distance = (TimeElapsed * 34300) / 2

    return distance

if name == '__main__':
    try:
        while True:
            dist = distance()
            print("Measured Distance = %.1f cm" % dist)
            time.sleep(1)
    except KeyboardInterrupt:
        print("Measurement stopped by User")
        GPIO.cleanup()
```

Results:



Button Interface to Raspberry Pi

Raspberry Pi is an ARM architecture processor based board designed for electronic engineers and hobbyists. The PI is one of most trusted project development platforms out there now. With higher processor speed and 1 GB RAM, the PI can be used for many high profile projects like Image processing and Internet of Things.

For doing any of high profile projects, one need to understand the basic functions of PI. That is why we are here, we will be covering all the **basic functionalities of Raspberry Pi** in these tutorials. In each tutorial series we will discuss one of functions of PI. By the end of tutorial series you will be able to do high profile projects by yourself.

Establishing **communication between PI and user** is very important for designing projects on PI. For the communication, PI must take Inputs from the user. In this second tutorial of PI series, we will **Interface a button to Raspberry Pi**, to take INPUTS from the user.

Here we will connect a button to one GPIO Pin and an LED to another GPIO pin of Raspberry Pi. We will write a program in PYTHON, to blink the LED continuously, on pressing the button by the user. LED will be blinking by turning the GPIO On and Off.

Before going for the programming, let's talk a bit about the LINUX and PYHTON.

LINUX:

LINUX is an Operating System like Windows. It performs all the basic functions which Windows OS can do. The main difference between them is, Linux is open source software where Windows is not. What it basically means is, Linux is free while Windows is not. Linux OS can be downloaded and operated for free, but for downloading genuine Windows OS, you have to pay the money.

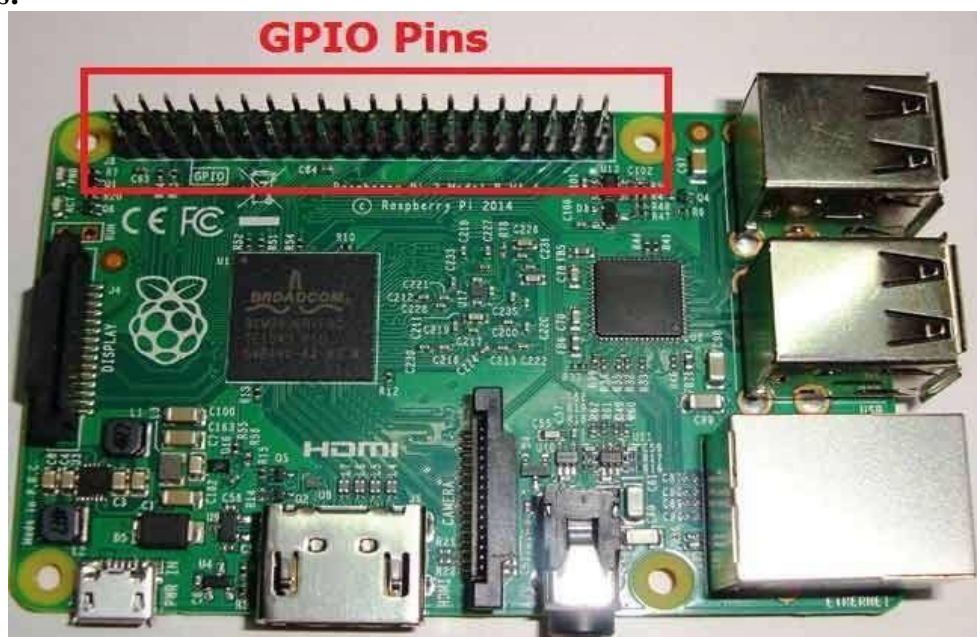
And another major difference between them is Linux OS can be „modified“ by tweaking into the code, but Windows OS cannot be modified, doing so will lead to legal complications. So anyone can take the Linux OS, and can modify it to his requirement in order to create his own OS. But we cannot do this in Windows, the Windows OS is provided with restrictions to stop you from editing OS.

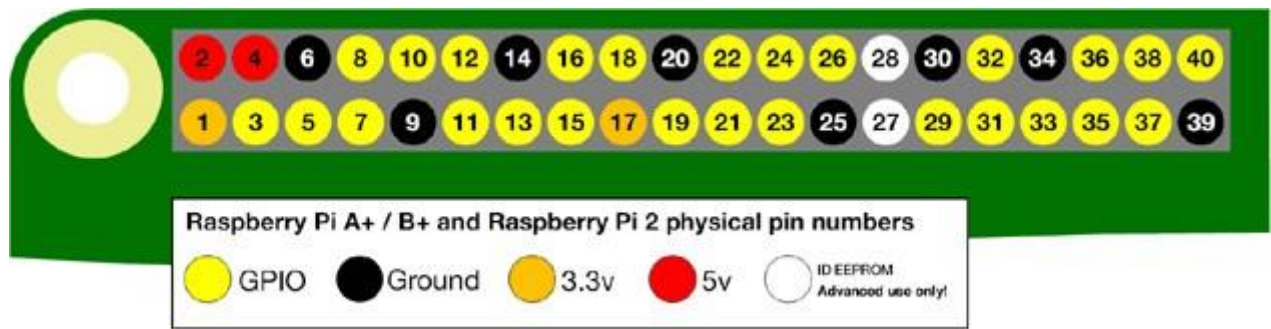
Here we are talking about Linux because, JESSIE LITE (Raspberry Pi OS) is LINUX based OS, which we have installed in Raspberry Pi Introduction part. The PI OS is generated on the grounds of LINUX, so we have to know a bit about LINUX operating commands. We will discuss about these Linux commands in the following tutorials.

PYTHON:

Unlike LINUX, **PYTHON is a programming language** like C, C++, and JAVA etc. These languages are used to develop applications. Remember programming languages run on Operating System. You cannot run a programming language without an OS. So OS is independent while programming languages are dependent. You can run PYTHON, C, C++, and JAVA on both Linux and Windows. Applications developed by these programming languages can be games, browsers, apps etc. We will use programming language PYTHON on our PI, to design projects and to manipulate the GPIO's. We will discuss a bit about PI GPIO before going any further,

GPIO Pins:





As shown in above figure, there are 40 output pins for the PI. But when you look at the second figure, you can see not all 40 pin out can be programmed to our use. These are only 26 GPIO pins which can be programmed. These pins go from **GPIO2 to GPIO27**.

These **26 GPIO pins can be programmed** as per need. Some of these pins also perform some special functions, we will discuss about that later. With special GPIO put aside, we have 17 GPIO remaining (Light green Cirl).

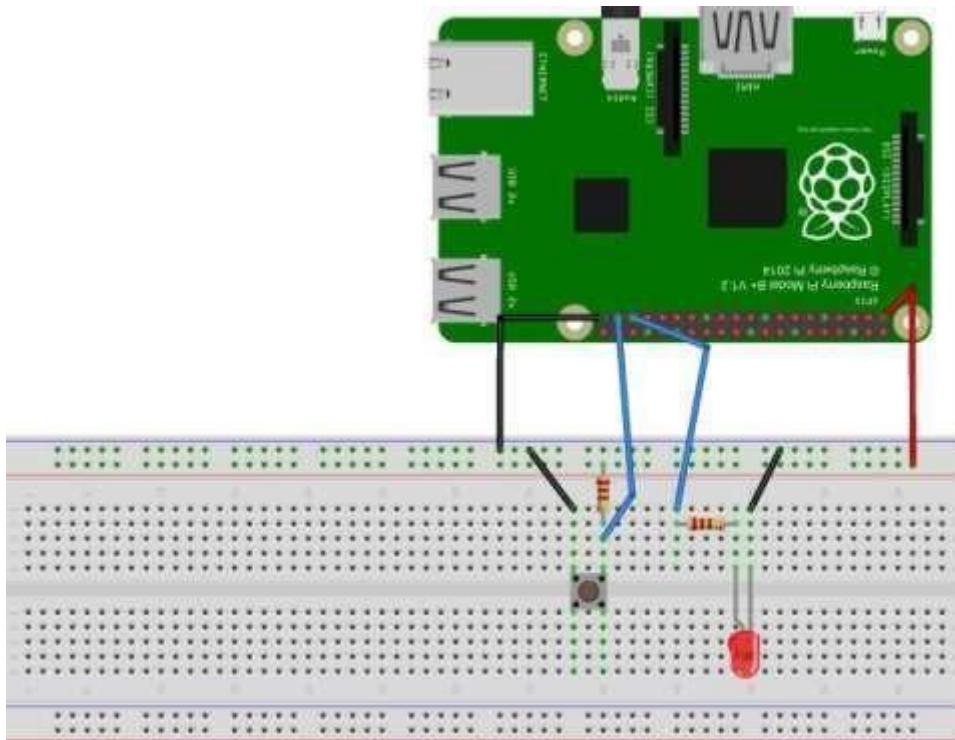
Each of these 17 GPIO pins can deliver a maximum of **15mA current**. And the sum of currents from all GPIO cannot exceed 50mA. So we can draw a maximum of 3mA in average from each of these GPIO pins. So one should not tamper with these things unless you know what you are doing.



Components Required:

Here we are using **Raspberry Pi 2 Model B with Raspbian Jessie OS**. All the basic Hardware and Software requirements are previously discussed, you can look it up in the Raspberry Pi Introduction, other than that we need:

1. Connecting pins
2. 220Ω or $1K\Omega$ resistor
3. LED
4. Button
5. Bread Board
6. Circuit Explanation:



As shown in the circuit diagram we are going to connect an LED to PIN35 (GPIO19) and a button to PIN37 (GPIO26). As said earlier, we cannot draw more than 15mA from any one of these pins, so to limit the current we are connecting a 220Ω or $1K\Omega$ resistor in series with the LED.

Working Explanation:

Once everything is connected, we can turn ON the Raspberry Pi to write the program in PYTHON and execute it. (To know how to use PYTHON go to PI BLINKY).

We will talk about few commands which we are going to use in PYTHON program. We are going to import GPIO file from library, below function enables us to program

GPIO pins of PI. We are also renaming “GPIO” to “IO”, so in the program whenever we want to refer to GPIO pins we will use the word „IO“.

```
import RPi.GPIO as IO
```

Sometimes, when the GPIO pins, which we are trying to use, might be doing some other functions. In that case, we will receive warnings while executing the program. Below command tells the PI to ignore the warnings and proceed with the program.

```
IO.setwarnings(False)
```

We can refer the GPIO pins of PI, either by pin number on board or by their function number. In pin diagram, you can see „PIN 37“ on the board is „GPIO26“. So we tell here either we are going to represent the pin here by „37“ or „26“.

```
IO.setmode (IO.BCM)
```

We are setting GPIO26 (or PIN37) as input pin. We will detect button press by this pin.

```
IO.setup(26,IO.IN)
```

While 1: is used for infinity loop. With this command the statements inside this loop will be executed continuously.

Once the program is executed, the LED connected to GPIO19 (PIN35) blinks whenever the button is pressed. Upon release the LED, it will go to OFF state again.

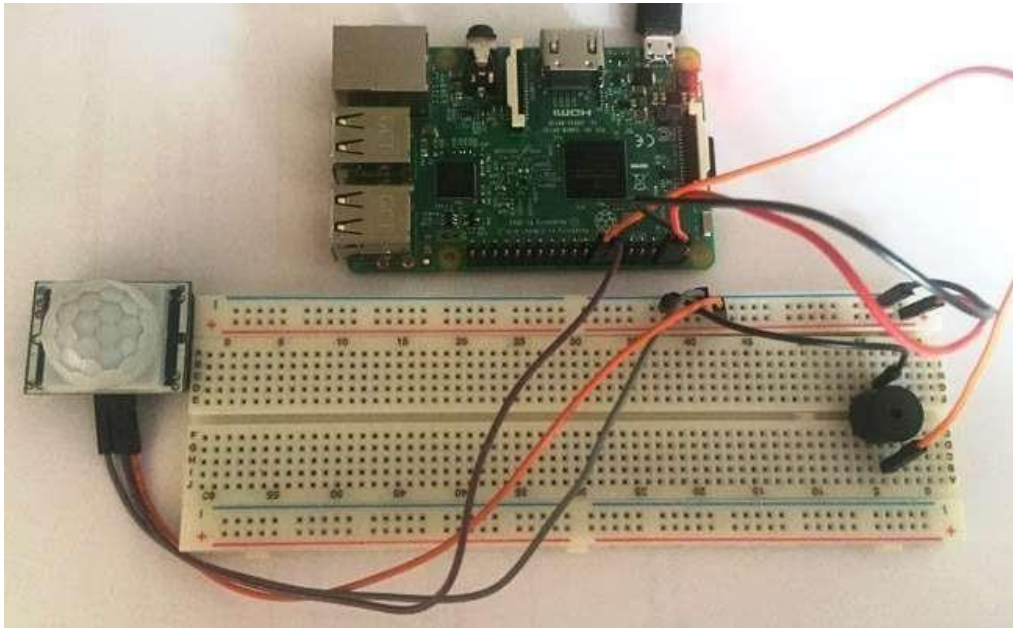
Code

```

import RPi.GPIO as IO
import time

#we are calling header file which helps us to use GPIO's of PI# we are
#do not show any warnings
#we are programming the GPIO by BCM pin numbers. (PIN39
#initialize GPIO19 as an output.#initialize GPIO26 as input
#execute loop forever
while 1:
    if IO.input(26) == False: #if GPIO26 goes low execute the below statements
        IO.output(19,True) # turn the LED on (making the voltage level HIGH)
        time.sleep(0.11) #sleep for 100m second
    else:
        IO.output(19,False) # turn the LED off (making GPIO19 low)
        time.sleep(1) #sleep for 100m second

```

Raspberry Pi Motion Sensor/Detector Circuit using PIR

Security systems play an important role in our day to day lives and there we can find a lot of different types of security systems with different kinds of technologies and with different price range. Being an electronic enthusiastic you can make a simple security system by spending few bucks and some spare time. Here in this article I am sharing a DIY guide to make a simple **Raspberry pi and PIR sensor based motion detector alarm** which will turn on the buzzer when the PIR sensor detects any human movement in the area. We also covered a simple PIR sensor based motion detector circuit in one of our previous articles where we covered the working of PIR sensor in detail.

Components Required

1. Raspberry Pi 3 (any model)
2. PIR Sensor
3. Buzzer
4. Breadboard
5. Connecting wires

Working of PIR sensor:

Passive Infrared (PIR) sensor is called passive because it receives infrared, not emits. Basically it detects any change in heat, and whenever it detects any change, its output PIN becomes HIGH. They are also referred as Pyroelectric or IR motion sensors. Here we should note that every object emits some amount of infrared when heated. Human also emits infrared because of body heat. **PIR sensors** can

detect small amount of variation in infrared. Whenever an object passes through the sensor range, it produces infrared because of the friction between air and object, and get caught by PIR.

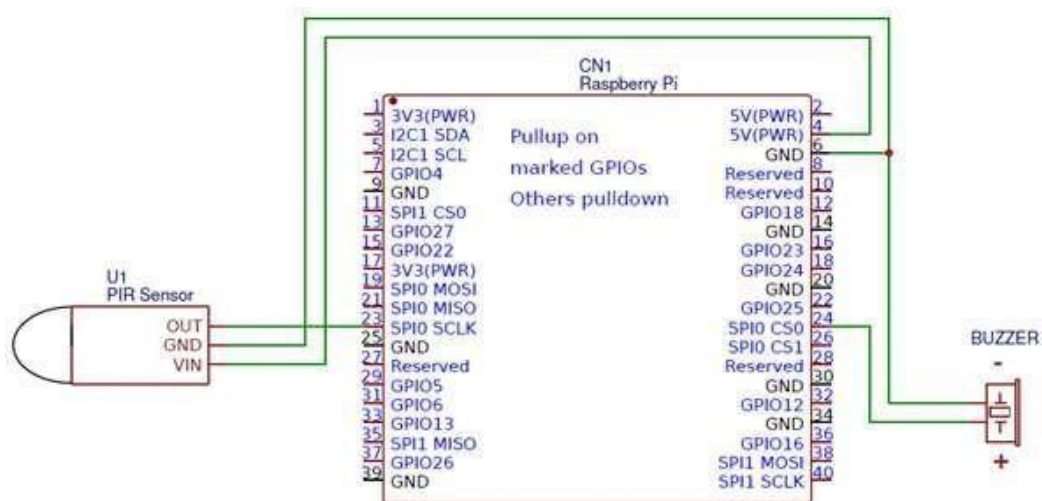
The main component of PIR sensor is **Pyroelectric sensor** shown in figure (rectangular crystal behind the plastic cap). Along with this, **BISS0001** ("Micro Power PIR Motion Detector IC"), some resistors, capacitors and other components used to build PIR sensor. BISS0001 IC take the input from sensor and does processing to make the output pin HIGH or LOW accordingly.



Pyroelectric sensor divide in two halves, when there is no motion, both halves remain in same state, means both senses the same level of infrared. As soon as somebody enters in first half, the infrared level of one half becomes greater than other, and this causes PIRs to react and makes the output pin high.

Pyroelectric sensor is covered by a plastic cap, which has array of many Fresnel Lens inside. These lenses are curved in such a manner so that sensor can cover a wide range.

Circuit Diagram for Raspberry Pi and PIR Sensor based Motion Detector



As shown in the above schematic diagram for Raspberry Pi and PIR sensor based motion detector, the positive pin of PIR sensor is connected with the pin 4 (5v) and ground pin of PIR sensor is connected with Pin 6 (Ground) of Raspberry Pi (You can find here the **Pin Diagram of Raspberry Pi**). The output pin of PIR sensor is connected with the GPIO 23 of Raspberry pi which is used to give input to Raspberry Pi. The GPIO pin 24 which is declared here for output is connected with positive of buzzer, and ground of buzzer is connected with the ground (pin 6) of raspberry pi.

Python Code for Raspberry Pi:

The Python code for this raspberry pi and PIR sensor based motion detector is quite simple and could be understood easily with the comments inline in the code section below. I declared the GPIO pin 23 and 24 as input and output pins.

```
while True:

    if GPIO.input(23): #If there is a movement, PIR sensor gives input to GPIO23
        GPIO.output(24, True) #Output given to Buzzer through GPIO24 time.sleep(1)
        #Buzzer turns on for 1 second
        GPIO.output(24, False)
```

A '*while*' loop is used as 'True' so the contents inside the loop will always execute. *if GPIO.input(23):* statement detects if GPIO pin 23 is high, and if the same is true it makes the output PIN 24 high. The function *time.sleep(secs)* is used to pause the program in python for particular time where 'secs' is the time in seconds. So here we used to pause it for 1 second. In the next line we made the output on 24 as false so buzzer stops until the loop begins the next iteration, as *While* loop is set always true without any pre-condition.

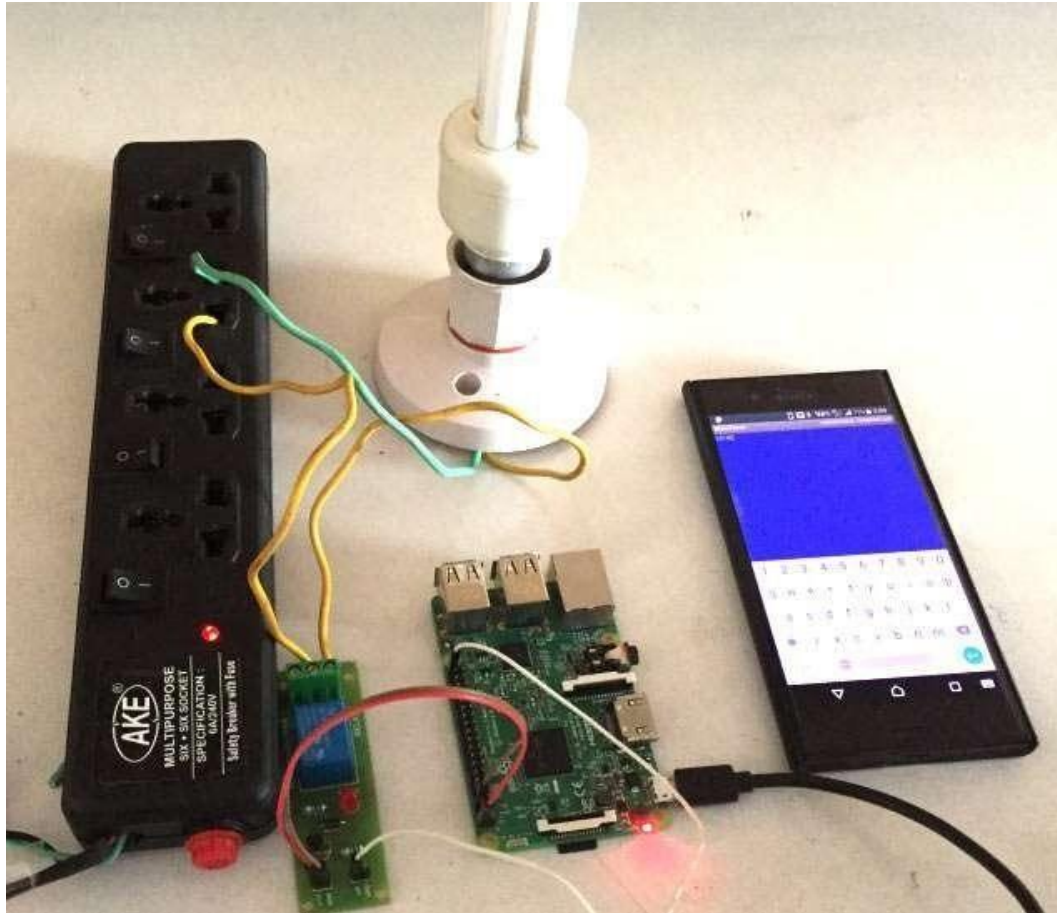
```
#Raspberry Pi Motion Detector Code
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.IN)
GPIO.setup(24, GPIO.OUT)
while True:

    if GPIO.input(23): #If there is a movement, PIR sensor gives input to GPIO 23
        GPIO.output(24, True) #Output given to Buzzer through GPIO 24
        time.sleep(1) #Buzzer turns on for 1 second
        GPIO.output(24, False)
        time.sleep(5)
    time.sleep(0.1)
```

RASPBERRY PI BASED SMART PHONE CONTROLLED HOMEAUTOMATION

EXPT. NO :

DATE :



Raspberry Pi based Smart Phone Controlled Home Automation

Raspberry Pi is very popular for IoT projects because of its seamless ability of wireless communication over internet. Raspberry Pi 3 has inbuilt Wi-Fi and Bluetooth, and Bluetooth is a very popular wireless communication Protocol. So today we are going to **Control Home Appliances through Smart Phone using Raspberry Pi 3 and Bluetooth**. Here we are using Raspberry Pi 3 which have inbuilt Bluetooth, so we don't need to use any external USB Bluetooth dongle. Apart from that we only need Relay Modules for this **Wireless Home Automation Project**. Here we are using RFCOMM Bluetooth protocol for wireless communication.

Programming for Bluetooth in Python follows the socket programming model and communications between the Bluetooth devices is done through RFCOMM socket. **RFCOMM** (Radio Frequency Communication) is a Bluetooth Protocol which provided emulated RS-232 serial ports and also called as Serial Port Emulation. Bluetooth serial port profile is based on this protocol. RFCOMM is very popular in Bluetooth applications because of its wide support and publically available API. It is bound to L2CAP protocol.

Installing Required Packages for Bluetooth Communication:

Before start, we need to install some softwares for **setting up Bluetooth communication in Raspberry Pi**. You should have a Raspbian Jessie installed memory card ready with Raspberry Pi. Check [this article](#) to install the Raspbian OS and getting started with Raspberry Pi. So now we first need to update the Raspbian using below commands:

```
sudo apt-get update
sudo apt-get upgrade
```

```
sudo apt-get install bluetooth blueman bluez
```

Then we need to install few Bluetooth related packages:

Then reboot the Raspberry Pi:

```
sudo reboot
```

BlueZ is a open source project and official Linux Bluetooth protocol stack. It supports all the core Bluetooth protocols and now become part of official Linux Kernel.

Blueman provides the Desktop interface to manage and control the Bluetooth devices. Finally we need **python Library for Bluetooth communication** so that we can send and receive data through RFCOMM using Python language:

```
sudo apt-get install python-bluetooth
```

Also install the GPIO support libraries for Raspberry Pi:

```
sudo apt-get install python-rpi.gpio
```

Now we are done with installing required packages for Bluetooth communication in Raspberry Pi.

Pairing Devices with Raspberry Pi over Bluetooth:

Pairing Bluetooth Devices, like mobile phone, with Raspberry Pi is very easy. Here we have **paired our Android Smart phone with Raspberry Pi**. We have previously installed BlueZ in Pi, which provides a command line utility called "***bluetoothctl***" to manage our Bluetooth devices.

Now open the *bluetoothctl* utility by below command:

```
sudo bluetoothctl
```

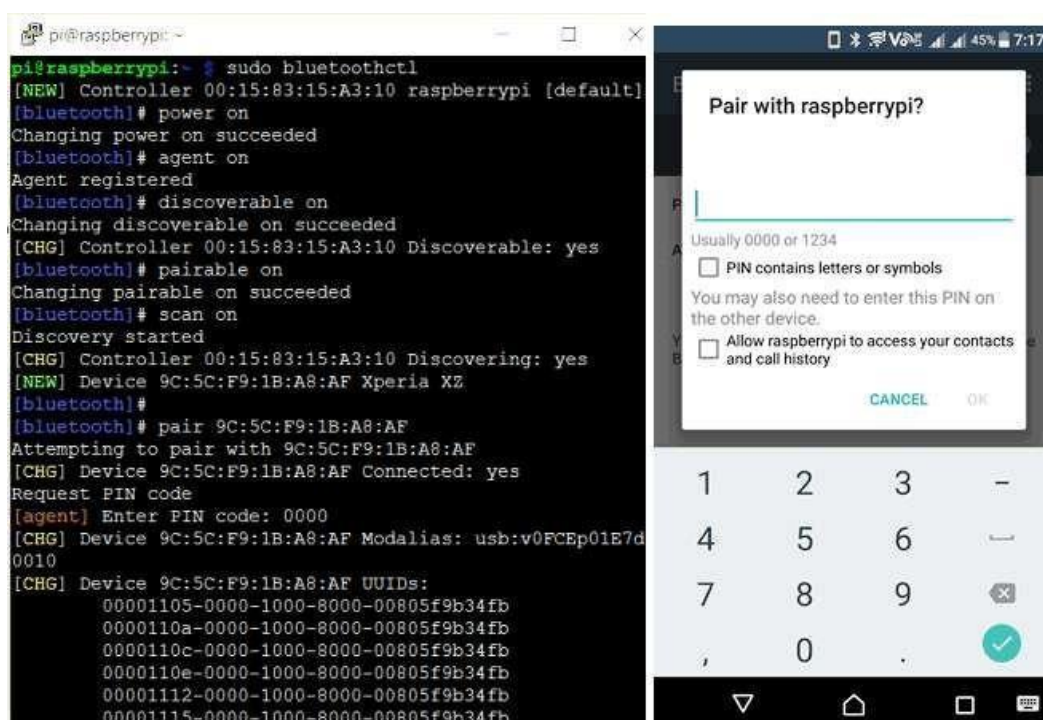
You can check all the commands of *bluetoothctl* utility by typing „*help*“. For now we need to enter below commands in given order:

```
[bluetooth]# power on
[bluetooth]# agent on
[bluetooth]# discoverable on
[bluetooth]# pairable on
[bluetooth]# scan on
```

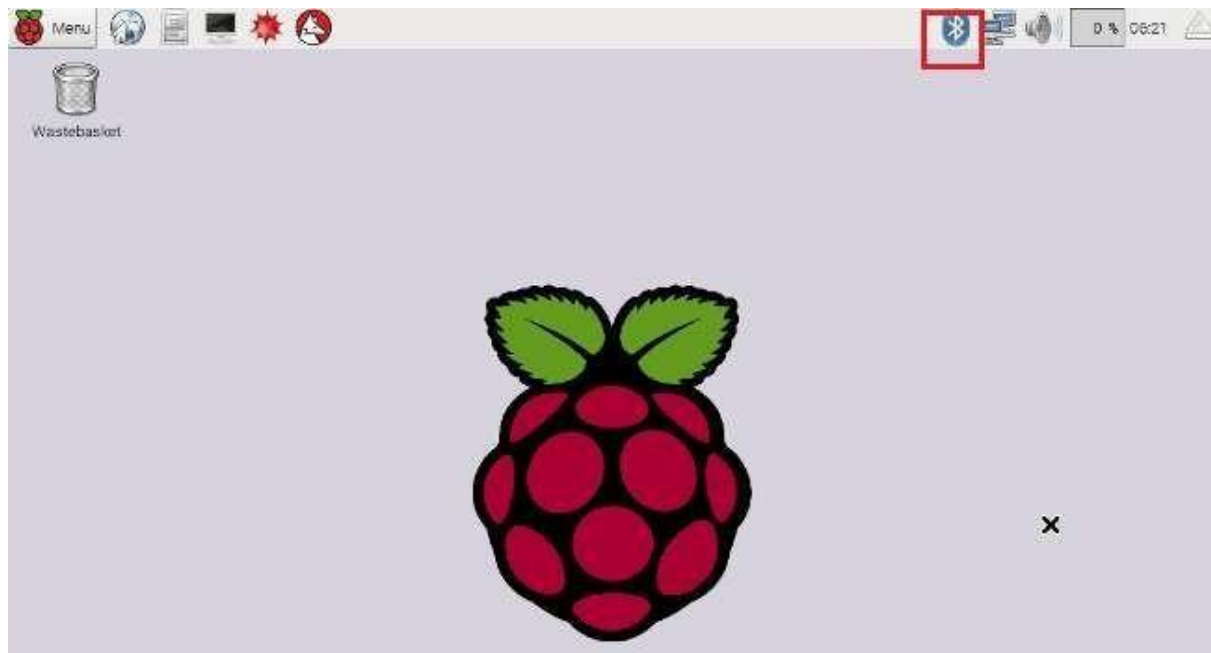
After the last command “*scan on*”, you will see your Bluetooth device (Mobile phone) in the list. Make sure that your mobile has Bluetooth turned on and visible by nearby devices. Then copy the MAC address of you device and pair it by using given command:

```
pair <address of your phone>
```

Then you will be prompted for Passcode or Pin in your Terminal console then type passcode there and press enter. Then type the same passcode in your mobile phone when prompted and you are now successfully paired with Raspberry Pi. We have also explained this whole process in the Video given in previous GPIO controlling Tutorial.

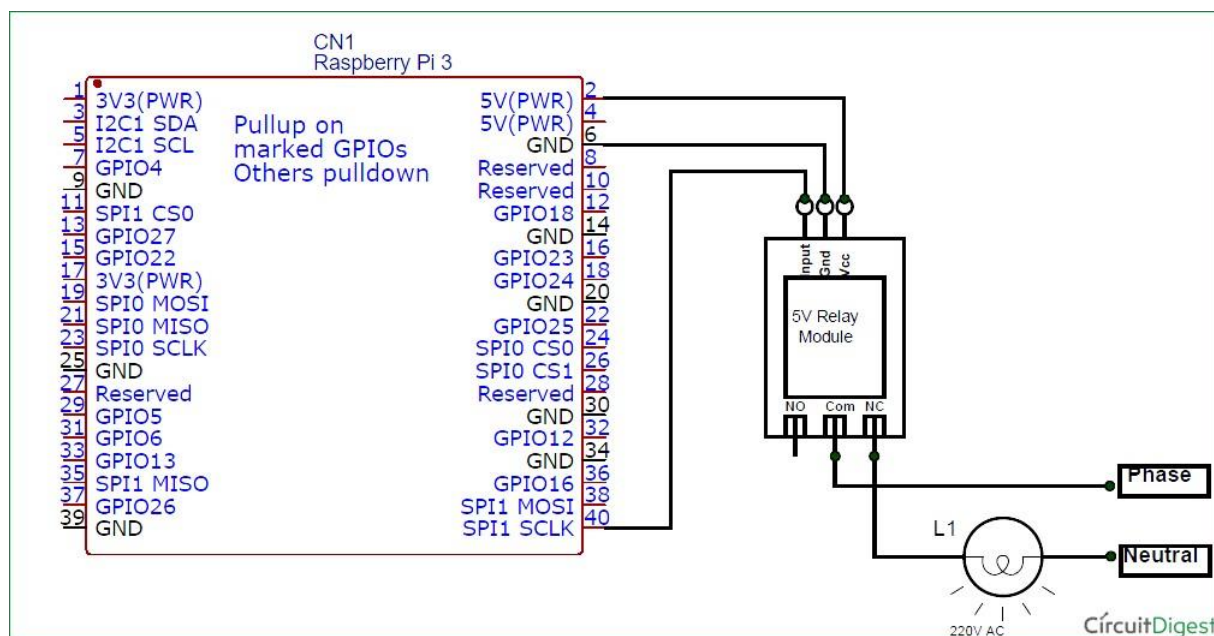


As told earlier, you can also use **Desktop interface to pair the Mobile phone**. After installing Blueman, you will see a Bluetooth icon in right side of your Raspberry Pi desktop as shown below, using which you can easily do the pairing.



Circuit Diagram:

Circuit diagram for this **Raspberry Pi based Bluetooth Controlled Home Automation** is very simple, we just connected Relay Module's input signal Pin to PIN 40 (GPIO 21) of Raspberry Pi and other two Pin (Vcc and GND of relay module) to Pin 2 and 6 of Raspberry Pi 3. Then we have connected a AC CFL bulb to the Relay as shown in the circuit diagram:



If you are new to Relay and want to learn more about Relay and its connections with ACappliance, check [this Article](#).



Relay Module

Controlling AC Appliance with Android App BlueTerm:

Now after pairing the Mobile Phone, we need to install a **Android App for communicating with Raspberry Pi using a Bluetooth Serial Adapter**, so that we can control the GPIO pins of Raspberry Pi. As told earlier RFCOMM/SPP protocol emulates serial communication over Bluetooth, so we installed here **BlueTerm App** which supports this protocol.



You can also use any other Bluetooth Terminal App which supports communication via RFCOMM socket.

Now after downloading and installing the BlueTerm App, **run the below given Python Program** from the terminal and connect the paired *raspberrypi* device from the BlueTerm App at the same time.



After successful connection you will see *connected:raspberrypi* at the top right corner of theApp as shown below:

```
pi@raspberrypi:~$ sudo apt-get install python-bl  
pi@raspberrypi:~$ python /home/pi/bluetooth_homea  
Accepted connection from ('9C:5C:F9:1B:A8:AF', 1)  
Received: 0  
AC light OFF  
Received: 1  
AC light ON  
Received: 0  
AC light OFF  
Received: 1  
AC light ON  
Received: 0  
AC light OFF  
Received: 1  
AC light ON  
Received: 0  
AC light OFF  
Received: 1  
AC light ON  
Received: 0  
AC light OFF  
Received: q  
Quit  
pi@raspberrypi:~$
```

BlueTerm connected: raspberrypi

101010101010101010101010q0101

Now you can just enter '1' or '0' from the BlueTerm app to make the GPIO pin HIGH and LOW respectively, which in turn triggers the Relay module connected to this pin, which further controls the AC bulb connected to Relay. Press 'q' to exit the program. You can use Google Voice Typing Keyboard to control the GPIO using your Voice. Check the complete demo in the Video given at the end.



So this is how you can **wirelessly control the AC Appliance using an Android App overBluetooth**.
Also check [How to use Bluetooth with Arduino](#).

Programming Explanation:

Python Program for **Controlling Raspberry Pi GPIO with Android App** is very simple and self-explanatory. Only we need to learn little bit about the code related to Bluetooth RFCOMM communication. First we need to import the Bluetooth socket library which enables us to control Bluetooth with Python language; we have installed the library for the same in the previous section.

```
import Bluetooth
```

Below is the code responsible for Bluetooth communication:

```
server_socket=bluetooth.BluetoothSocket( bluetooth.RFCOMM )

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

client_socket,address = server_socket.accept()
print "Accepted connection from ",address while
1:
    data = client_socket.recv(1024)
```

Here we can understand them line by line:

server_socket=bluetooth.BluetoothSocket(bluetooth.RFCOMM): Creating socket for Bluetooth RFCOMM communication.

server_socket.bind(("", port)):- Server binds the script on host "" to port.

server_socket.listen(1): Server listens to accept one connection at a time.

client_socket, address = server_socket.accept(): Server accepts client's connection request and assign the mac address to the variable *address*, *client_socket* is the client's socket

data = client_socket.recv(1024): Receive data through the client socket *client_socket* and assign it to the variable *data*. Maximum 1024 characters can be received at a time.

Finally after all the programming, close the client and server connection using below code:

```
client_socket.close()
server_socket.close()
```

All the other code is easy and self-explanatory. Check the **full code below**. Try to modify this project and you can use it to control many other things wirelessly, [Robot car through android phone](#) or can use your [voice to control the lights](#).

Also check our many types of **Home Automations Projects using different technologies and Microcontrollers** like:

1. [DTMF Based Home Automation](#)
2. [GSM Based Home Automation using Arduino](#)
3. [PC Controlled Home Automation using Arduino](#)
4. [Bluetooth Controlled Home Automation using 8051](#)
5. [IR Remote Controlled Home Automation using Arduino](#)
6. [home automation project using MATLAB and Arduino](#)
7. [RF Remote Controlled LEDs Using Raspberry Pi](#)
8. [Smart Phone Controlled Home Automation using Arduino](#)
9. [Voice Controlled Home Automation using ESP8266 and Android App](#)
10. [RF based Home Appliances System without Microcontroller](#)

Program:

```
import bluetooth

import RPi.GPIO as GPIO #calling for header file which helps in using GPIOs of Pi
GPIO.setmode(GPIO.BCM)      #programming the GPIO by BCM pin numbers. (like PIN40 as
GPIO21)
GPIO.setwarnings(False)
GPIO.setup(BULB,GPIO.OUT) #initialize GPIO21 (Relay connected at this pin) as an output Pin
GPIO.output(BULB,0)

server_socket=bluetooth.BluetoothSocket( bluetooth.RFCOMM )port = 1
server_socket.bind(("",port))
server_socket.listen(1)

client_socket,address = server_socket.accept()
print "Accepted connection from ",address while 1:
```

```
data = client_socket.recv(1024)
print "Received: %s" % data
if (data == "0"): #if '0' is sent from the Android App, turn OFF the CFL bulbprint ("AC
    light OFF")
    GPIO.output(BULB,0)

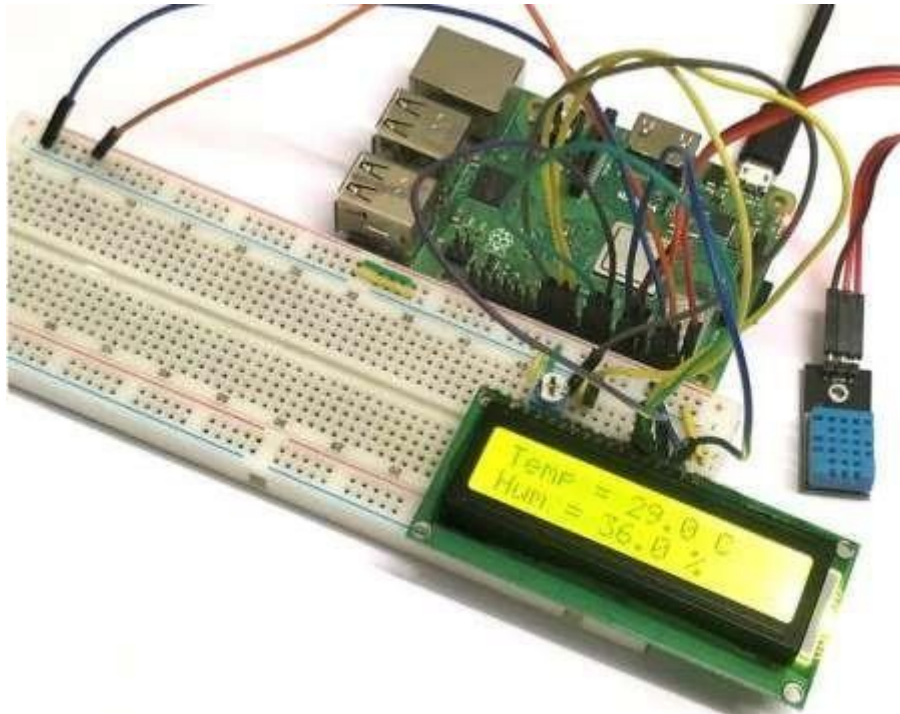
if (data == "1"): #if '1' is sent from the Android App, turn OFF the CFL bulbprint ("AC
    light ON")
    GPIO.output(BULB,1)if
(data == "q"):
    print ("Quit")
    break

client_socket.close()
server_socket.close()
```


INTERFACING DHT11 TEMPERATURE AND HUMIDITY SENSOR WITH RASPBERRY PI

EXPT. NO :

DATE :



Interfacing DHT11 with Raspberry Pi

Temperature and Humidity are the most common parameters that are being monitored in any environment. There are tons of sensors to choose from for **measuring temperature and humidity**, but the most used one is the **DHT11** due to its decent measuring range and accuracy. It also works with one pin communication and hence is very easy to interface with Microcontrollers or Microprocessors. In this tutorial we are going to **learn how to interface the popular DHT11 sensor with Raspberry Pi** and display the value of temperature and humidity on a 16x2 LCD screen. We already used it to build IoT Raspberry Pi Weather Station.

Overview of DHT11 Sensor:

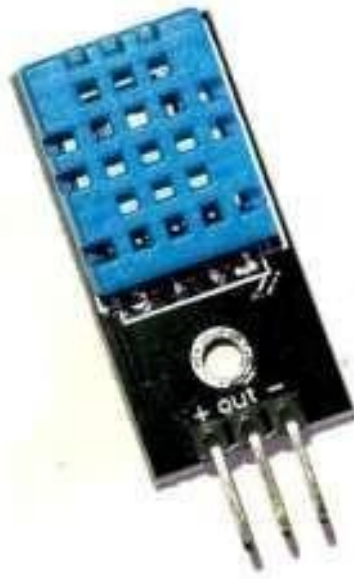
The DHT11 sensor can measure relative humidity and temperature with the following specifications

Temperature Range: 0-50°C

Temperature Accuracy: ± 2 °C

Humidity Range: 20-90% RH

Humidity Accuracy: ± 5 %



DHT11 Temperature and Humidity Sensor

The DHT11 sensor is available either in module form or in sensor form. In this tutorial we are using the module form of the sensor, the only difference between both is that in module form the sensor has a filtering capacitor and a pull up resistor attached to the output pin of the sensor. So if you are using the sensor alone make sure you add these two components. Also learn [DHT11 interfacing with Arduino](#).

How DHT11 Sensor works:

The DHT11 sensor comes with a blue or white colour casing. **Inside this casing we have two important components** which help us to sense the relative humidity and temperature. **The first component is a pair of electrodes**; the electrical resistance between these two electrodes is decided by a moisture holding substrate. So the measured resistance is inversely proportional to the relative humidity of the environment. Higher the relative humidity lower will be the value of resistance and vice versa. Also note that Relative humidity is different from actual humidity. Relative humidity measures the water content in air relative to the temperature in the air.

The other component is a surface mounted NTC Thermistor. The term NTC stands for Negative temperature coefficient, for increase in temperature the value of resistance will decrease

Pre-Requisites:

It is assumed that your Raspberry Pi is already flashed with an operating system and is able to connect to the internet.

It is also assumed that you have access to your pi either through terminal windows or through other application through which you can write and execute python programs and use the terminal window.

Installing the Adafruit LCD library on Raspberry Pi:

The value of the temperature and humidity will be displayed on a 16*2 LCD display. Adafruit provides us a library to easily operate this LCD in 4-bit mode, so let us add it to our Raspberry Pi by opening the terminal window Pi and following the below steps.

Step 1: Install git on your Raspberry Pi by using the below line. **Git** allows you to clone any project files on Github and use it on your Raspberry pi. Our library is on Github so we have to install git to download that library into pi.

```
apt-get install git
```

Step 2: The following line links to the GitHub page where the library is present just execute the line to clone the project file on Pi home directory

```
git clone git://github.com/adafruit/Adafruit_Python_CharLCD
```

Step 3: Use the below command to change directory line, to get into the project file that we just downloaded. The command line is given below

```
cd Adafruit_Python_CharLCD
```

Step 4: Inside the directory there will be a file called *setup.py*, we have to install it, to install the library. Use the following code to install the library

```
sudo python setup.py install
```

That is it **the library should have been installed successfully**. Now similarly let's proceed with installing the DHT library which is also from Adafruit.

Installing the Adafruit DHT11 library on Raspberry Pi:

DHT11 Sensor works with the principle of one-wire system. The value of temperature and humidity is sensed by the sensor and then transmitted through the output pin as serial data. We can then read these data by using I/O pin on a MCU/MPU. To understand how these values are read you would have to read through the **datasheet** of the DHT11 sensor, but for now to keep things simple we will use a library to talk with the DHT11 sensor.

The **DHT11 library** provided by Adafruit can be used for DHT11, DHT22 and other one wire temperature sensors as well. The procedure to install the DHT11 library is also similar to the one followed for installing LCD library. The only line that would change is the link of the GitHub page on which the DHT library is saved.

Enter the four command lines one by one on the terminal to **install the DHT library**

```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
```

```
cd Adafruit_Python_DHT
```

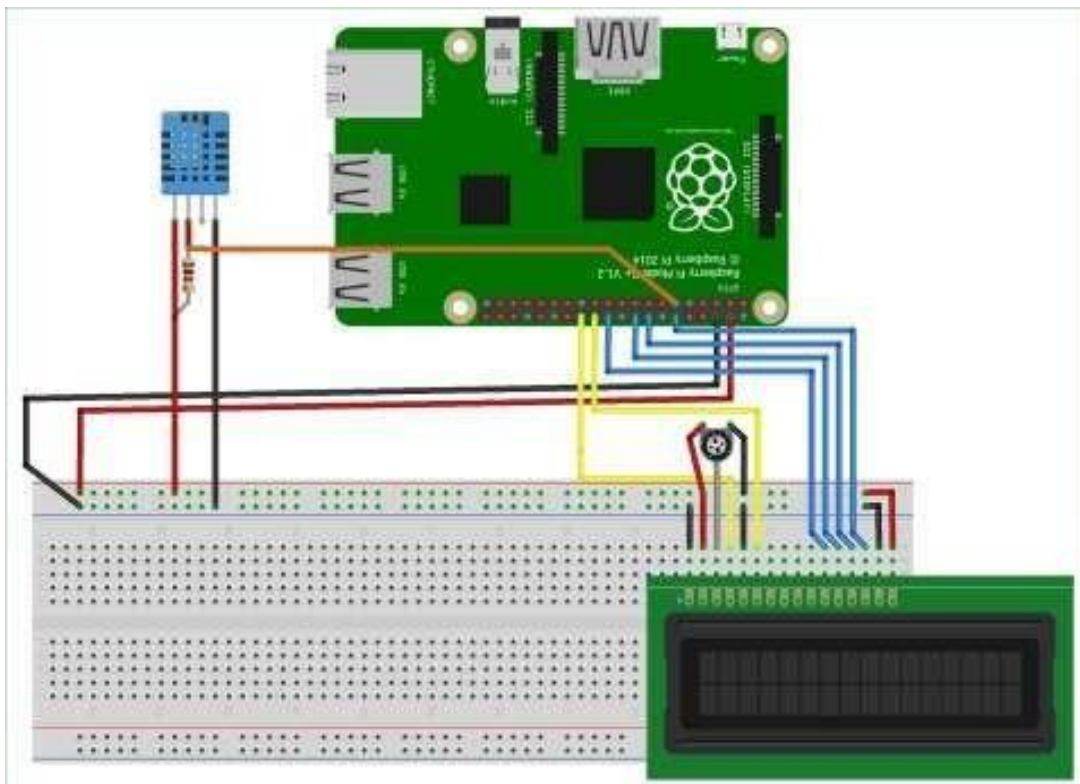
```
sudo apt-get install build-essential python-dev
```

```
sudo python setup.py install
```

Once it is done you will have both the libraries successfully installed on our Raspberry Pi. Now we can proceed with the hardware connection.

Circuit Diagram:

The complete circuit diagram **Interfacing DH11 with Raspberry pi** is given below, it was built using Fritzing. Follow the connections and make the circuit

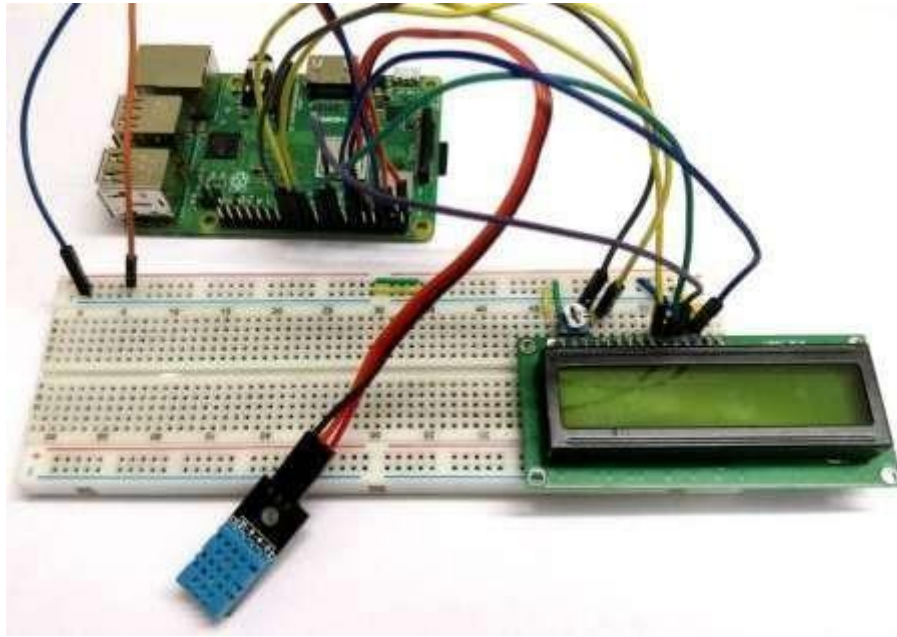


Both the **LCD and DHT11 sensor** works with +5V supply so we use the 5V pins on the Raspberry Pi to power both. A pull up resistor of value 1k is used on the output pin of the DHT11 sensor, if you are using a module you can avoid this resistor.

A **trimmer pot of 10k** is added to the Vee pin of the LCD to control the contrast level of the LCD. Other than that all the connections are pretty straight forward. But make a note of which GPIO pins you are using to connect the pins since we will need in our program. The below chart should allow you to figure out the GPIO pin numbers.



Use the chart and make your connections according to the circuit diagram. I used a breadboard and jumper wires to make my connections. Since, I used DHT11 module I wired it directly to Raspberry Pi. My hardware looked like this below



Python Programming for DHT11 sensor:

We have to write a program to **read the value of temperature and humidity from the DHT11 sensor** and then display the same on the LCD. Since we have downloaded libraries for both LCD and DHT11 sensor the code should be pretty much straight forward. The **python complete program** can be found at the end of this page, but you can read further to understand how the program works.

We have to **import the LCD library and DHT11 library** into our program to use the functions related to it. Since we have already downloaded and installed them on our Pi we can simply use the following lines to import them. We also **import the time library** to use the delay function.

```
import time #import time for creating delay
import Adafruit_CharLCD as LCD #Import LCD library
import Adafruit_DHT #Import DHT Library for sensor
```

Next, we have to **specify to which pins the sensor is connected** to and what type of temperature sensor is used. The variable *sensor_name* is assigned to `Adafruit_DHT.DHT11` since we are using the DHT11 sensor here. The output pin of the sensor is connected to GPIO 17 of the Raspberry Pi and hence we assign 17 to *sensor_pin* variable as shown below.

```
sensor_name = Adafruit_DHT.DHT11 #we are using the DHT11 sensor
sensor_pin = 17 #The sensor is connected to GPIO17 on Pi
```

Similarly, we also have to **define to which GPIO pins the LCD is connected to**.

Here we are using the **LCD in 4-bit mode** hence we will have four data pins and two control pins to connect to the GPIO pins of the pi. Also, you can connect the backlight pin to a GPIO pin if we wish to control the backlight also. But for now I am not using that so I have assigned 0 to it.

```
lcd_rs    = 7 #RS of LCD is connected to GPIO 7 on PI
lcd_en    = 8 #EN of LCD is connected to GPIO 8 on PI
lcd_d4    = 25 #D4 of LCD is connected to GPIO 25 on PI
lcd_d5    = 24 #D5 of LCD is connected to GPIO 24 on PI
lcd_d6    = 23 #D6 of LCD is connected to GPIO 23 on PI
lcd_d7    = 18 #D7 of LCD is connected to GPIO 18 on PI
lcd_backlight = 0 #LED is not connected so we assign to 0
```

You can also connect LCD in 8-bit mode with Raspberry pi but then free pins will be reduced.

The LCD library from Adafruit that we downloaded can be used for all types of characteristic LCD displays. Here in our project we are using a 16*2 LCD display so we are mentioning the number of Rows and Columns to a variable as shown below.

```
lcd_columns = 16 #for 16*2 LCD
lcd_rows    = 2 #for 16*2 LCD
```

Now, that we have declared the LCD pins and the number of Rows and Columns for the LCD we can **initialize the LCD display** by using the following line which sends all the required information to the library.

```
lcd = LCD.Adafruit_CharLCD(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7,
lcd_columns, lcd_rows, lcd_backlight) #Send all the pin details to library
```

To start the program, we **display a small intro message** using the `lcd.message()` function and then give a delay of 2 second to make the message readable. For printing on the 2nd line the command `\n` can be used as shown below

```
lcd.message('DHT11 with Pi \n -CircuitDigest') #Give a intro message
time.sleep(2) #wait for 2 secs
```


Finally, **inside our *while* loop we should read the value of temperature and humidity** from the sensor and display it on the LCD screen for every 2 seconds. The complete program inside the while loop is shown below

while 1: #Infinite Loop

humidity, temperature = Adafruit_DHT.read_retry(sensor_name, sensor_pin) #read from sensor and save respective values in temperature and humidity varibale

lcd.clear() #Clear the LCD screen

lcd.message ('Temp = %.1f C' % temperature) # Display the value of temperature

lcd.message ('\nHum = %.1f %%' % humidity) #Display the value of Humidity

time.sleep(2) #Wait for 2 sec then update the values

We can easily get the value of temperature and humidity from the sensor using this single line below. As you can see it return two values which is stored in the **variable humidity and temperature**. The *sensor_name* and *sensor_pin* details are passed as parameters; these values were updated in the beginning of the program

humidity, temperature = Adafruit_DHT.read_retry(sensor_name, sensor_pin)

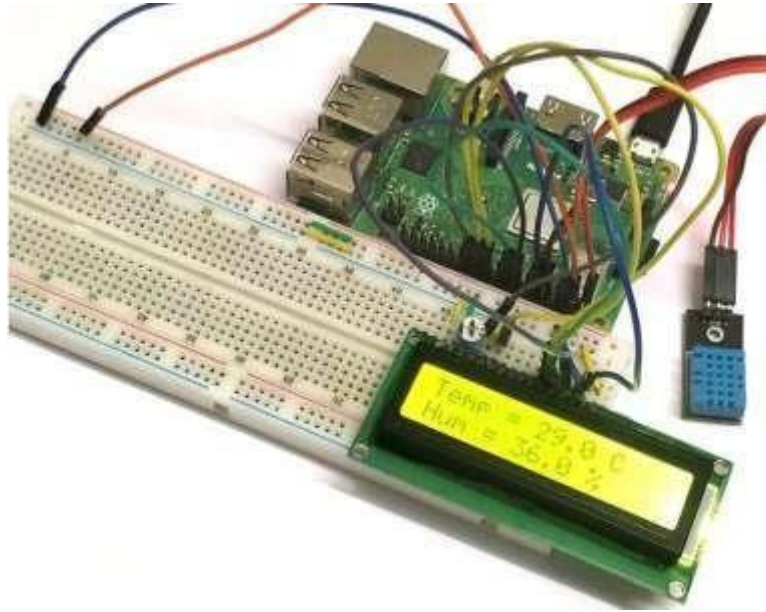
To display a variable name on the LCD screen we can use the identifiers like &d, %c etc. Here since we are displaying a floating point number with only one digit after the decimal point we use the identifier %.1f for **displaying the value in the variable temperature and humidity**

lcd.message ('Temp = %.1f C' % temperature)

lcd.message ('\nHum = %.1f %%' % humidity)

Measuring Humidity and Temperature using Raspberry Pi:

Make the connections as per the circuit diagram and install the required libraries. Then launch the python program given at the end of this page. Your LCD should display an intro message and then display the current temperature and humidity value as shown in the image below.



If you find nothing being displayed the LCD, check if the python shell window is displaying any errors, if no error is displayed then check your connections once more and adjust the potentiometer to vary the contrast level of the LCD and check if you get anything on the screen.

Hope you understood the project and enjoyed building it, if you has faced any problem in getting this done report it on the comment section or use the forum for technical help. I will try my best to respond to all comments.

You can also check our other projects using DHT11 with other microcontroller.

Code

```
#Program to read the values of Temp and Hum from the DHT11 sensor and display them on the LCD
#Website: www.circuitdigest.com
```

```
import time #import time for creating delay
```

```
import Adafruit_CharLCD as LCD #Import LCD libraryimport
```

```
Adafruit_DHT #Import DHT Library for sensor
```

```
sensor_name = Adafruit_DHT.DHT11 #we are using the DHT11 sensor sensor_pin = 17
```

```
#The sensor is connected to GPIO17 on Pi
```

```
lcd_rs      = 7 #RS of LCD is connected to GPIO 7 on PI lcd_en
```

```
            = 8 #EN of LCD is connected to GPIO 8 on PI lcd_d4
```

```
            = 25 #D4 of LCD is connected to GPIO 25 on PI
```

```
lcd_d5      = 24 #D5 of LCD is connected to GPIO 24 on PI
```

```
lcd_d6      = 23 #D6 of LCD is connected to GPIO 23 on PI
```

```
lcd_d7      = 18 #D7 of LCD is connected to GPIO 18 on PI lcd_backlight = 0 #LED is not
connected so we assign to 0 lcd_columns = 16 #for 16*2 LCD
```

```
lcd_rows    = 2 #for 16*2 LCD
```

```
lcd = LCD.Adafruit_CharLCD(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, lcd_columns, lcd_rows,
lcd_backlight) #Send all the pin details to library
```

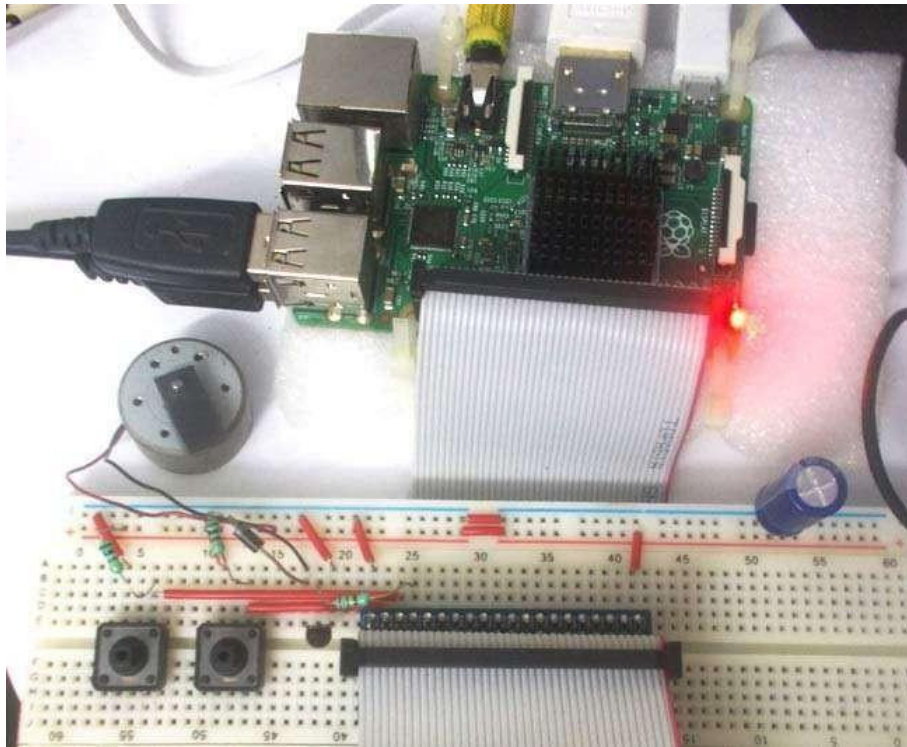
```
lcd.message('DHT11 with Pi \n -CircuitDigest') #Give a intro message time.sleep(2) #wait for 2 secs
```

```
while 1: #Infinite Loop
```

```
    humidity, temperature = Adafruit_DHT.read_retry(sensor_name, sensor_pin) #read from sensor and save
    respective values in temperature and humidity variable
```

```
    lcd.clear() #Clear the LCD screen
```

```
    lcd.message('Temp = %.1f C' % temperature) # Display the value of temperature lcd.message('\nHum =
    %.1f %%' % humidity) #Display the value of Humidity time.sleep(2) #Wait for 2 sec then update the
    values
```

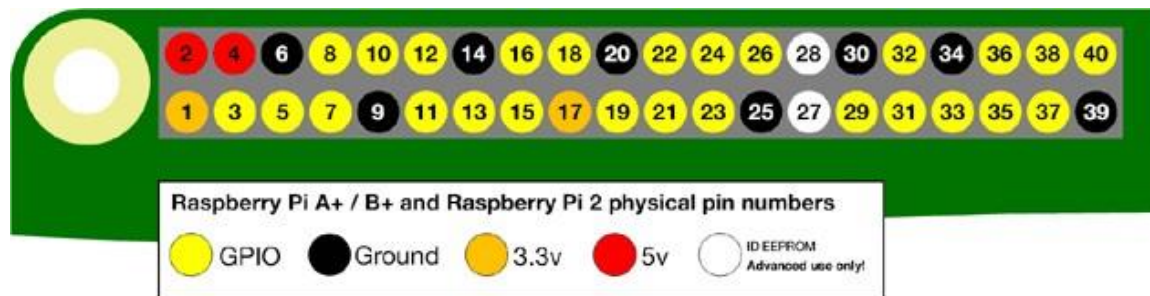


DC Motor Speed Control with Raspberry Pi

Raspberry Pi is an ARM architecture processor based board designed for electronic engineers and hobbyists. The PI is one of most trusted project development platforms out there now. With higher processor speed and 1 GB RAM, the PI can be used for many high profile projects like Image processing and Internet of Things.

For doing any of high profile projects, one need to understand the basic functions of PI. We will be covering all the **basic functionalities of Raspberry Pi** in these tutorials. In each tutorial we will discuss one of functions of PI. By the end of tutorial series you will be able to do high profile projects by yourself. Check these for **Getting Started with RaspberryPi** and **Raspberry Pi Configuration**.

We have discussed **LED Blinky**, **Button Interfacing** and **PWM generation** in previous tutorials. In this tutorial we will **Control the Speed of a DC motor using Raspberry Pi** and **PWM** technique. **PWM** (Pulse Width Modulation) is a method used for getting variable voltage out of constant power source. We have discussed about PWM in the previous tutorial. There are **40 GPIO output pins in Raspberry Pi 2**. But out of 40, only 26 GPIO pins (GPIO2 to GPIO27) can be programmed. Some of these pins perform some special functions. With special GPIO put aside, we have 17 GPIO remaining. To know more about GPIO pins,



Each of these 17 GPIO pin can deliver a maximum of **15mA**. And the sum of currents from all GPIO Pins cannot exceed **50mA**. So we can draw a maximum of 3mA in average from each of these GPIO pins. So one should not tamper with these things unless you know what you are doing.

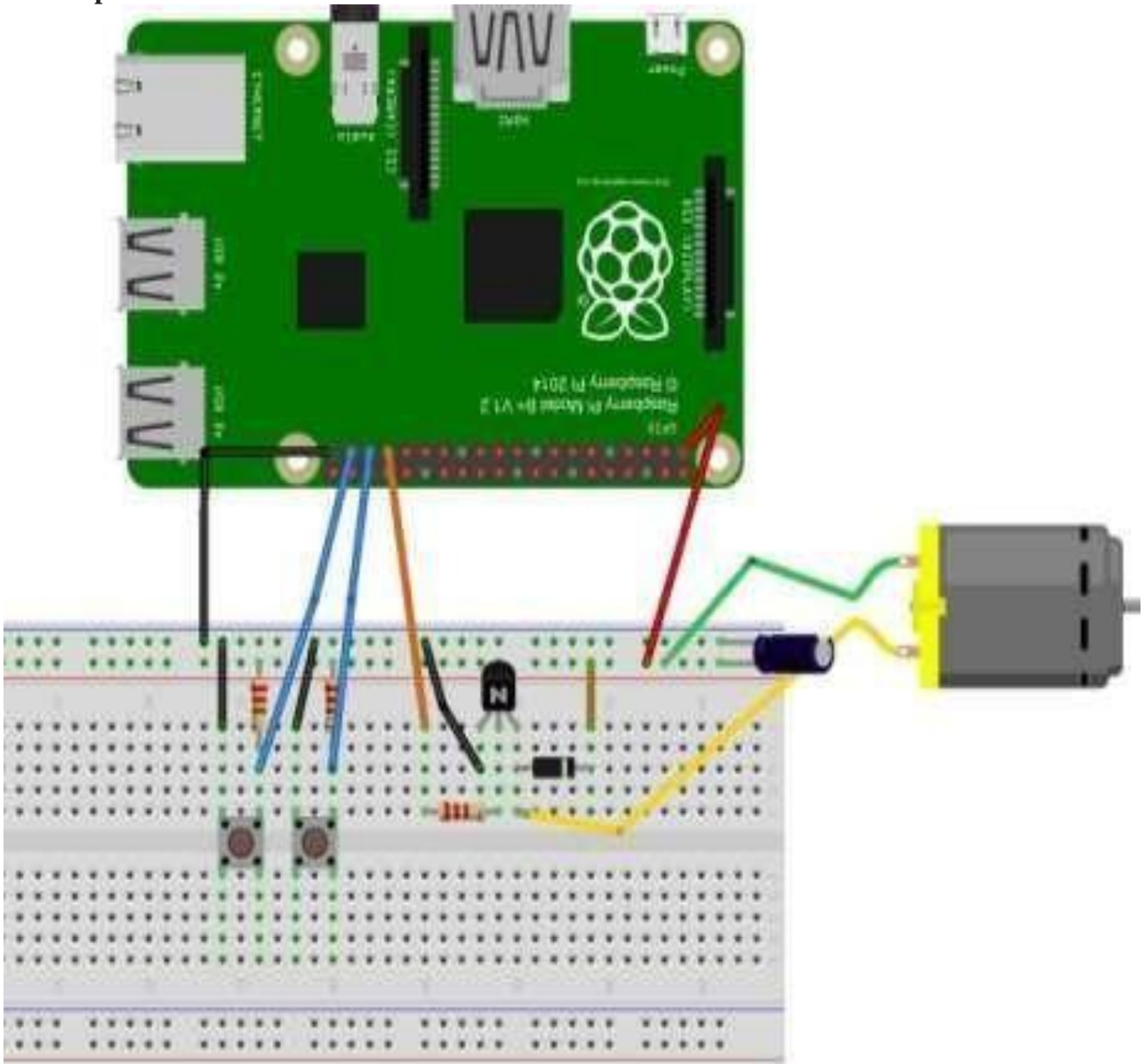
There are **+5V (Pin 2 & 4) and +3.3V (Pin 1 & 17) power output pins** on the board for connecting other modules and sensors. This power rail is connected in parallel to processor power. So drawing High current from this power rail affects the Processor. There is a fuse on the PI board which will trip once you apply high load. You **can draw 100mA safely from the +3.3V** rail. We are talking about this here because; we are connecting the DC motor to +3.3V. With the power limit in mind, we can only connect low power motor here, if you want to drive high power motor, consider powering it from a separate power source.

Components Required:

Here we are using **Raspberry Pi 2 Model B with Raspbian Jessie OS**. All the basic Hardware and Software requirements are previously discussed, you can look it up in the **Raspberry Pi Introduction**, other than that we need:

1. Connecting pins
2. 220Ω or 1KΩ resistor (3)
3. Small DC Motor
4. Buttons (2)
5. 2N2222 Transistor
6. 1N4007 Diode
7. Capacitor- 1000uF
8. Bread Board

Circuit Explanation:



As said earlier, we **cannot draw more than 15mA** from any GPIO pins and DC motor draws more than 15mA, so the PWM generated by Raspberry Pi cannot be fed to the DC motor directly. So if we connect the motor directly to PI for speed control, the board might get damaged permanently.

So we are going to use an **NPN transistor (2N2222) as a switching device**. This transistor here drives the high power DC motor by taking PWM signal from PI. Here one should pay attention that wrongly connecting the transistor might load the board heavily.

The motor is an induction and so while switching the motor, we experience inductive spiking. This spiking will heat up the transistor heavily, so we will be **using Diode (1N4007)** to provide protection to transistor **against Inductive Spiking**.

In order to **reduce the voltage fluctuations**, we will be connecting a **1000uF capacitor** across the power supply as shown in the Circuit Diagram.

Working Explanation:

Once everything is connected as per the circuit diagram, we can turn ON the PI to write the program in PYHTON.

We will talk about few commands which we are going to use in **PYHTON program**. We are going to import GPIO file from library, below function enables us to program

GPIO pins of PI. We are also renaming “GPIO” to “IO”, so in the program whenever we want to refer to GPIO pins we will use the word „IO“.

```
import RPi.GPIO as IO
```

Sometimes, when the GPIO pins, which we are trying to use, might be doing some other functions. In that case, we will receive warnings while executing the program. Below command tells the PI to ignore the warnings and proceed with the program.

```
IO.setwarnings(False)
```

We can refer the GPIO pins of PI, either by pin number on board or by their function number. Like „PIN 35“ on the board is „GPIO19“. So we tell here either we are going to represent the pin here by „35“ or „19“.

```
IO.setmode (IO.BCM)
```

```
IO.setup(19,IO.IN)
```

We are setting GPIO19 (or PIN35) as output pin. We will get PWM output from this pin. After setting the pin as output we need to setup the pin as PWM output pin,

```
p = IO.PWM(output channel , frequency of PWM signal)
```

The above command is for setting up the channel and also for setting up the frequency of the PWM signal. „p“ here is a variable it can be anything. We are using GPIO19 as the PWM *output channel*. „frequency of PWM signal“ has been chosen 100, as we don't want to see LED blinking.

Below command is used to start PWM signal generation, „DUTYCYCLE“ is for setting the Turn On ratio, 0 means LED will be ON for 0% of time, 30 means LED will be ON for 30% of the time and 100 means completely ON.

```
p.start(DUTYCYCLE)
```

In case the Condition in the braces is true, the statements inside the loop will be executed once. So if the GPIO pin 26 goes low, then the statements inside the IF loop will be executed once. If the GPIO pin 26 does not go low, then the statements inside the IF loop will not be executed.

While 1: is used for infinity loop. With this command the statements inside this loop will be executed continuously.

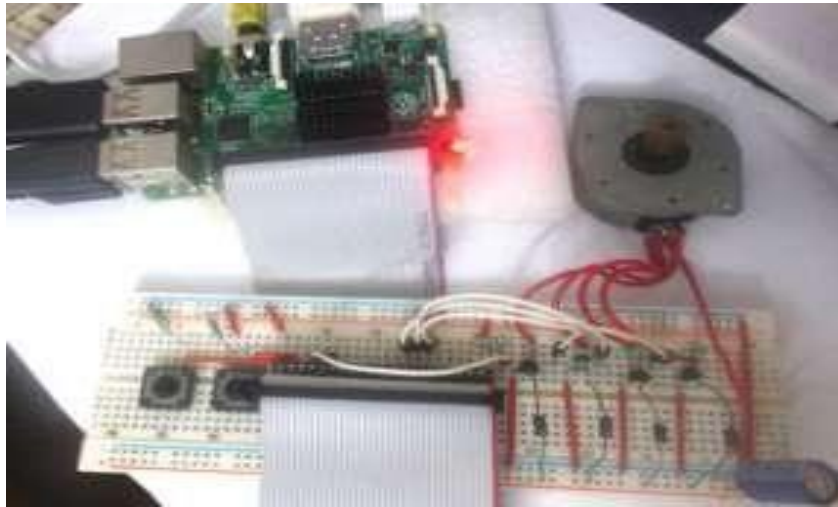
```
if(IO.input(26) == False):
```

We have all the commands needed to achieve the speed control with this.

After writing the program and executing it, all there is left is operating the control. We have two buttons connected to PI; one for **incrementing the Duty Cycle** of PWM signal and other for **decrementing the Duty Cycle** of PWM signal. By pressing one button the, speed of DC motor increases and by pressing the other button, the speed of DC motor decreases. With this we have achieved the **DC Motor Speed Control by Raspberry Pi**.

Program

```
import RPi.GPIO as IO      # calling header file which helps us use GPIO's of PI
import time                # calling time to provide delays in program
IO.setwarnings(False)     #do not show any warnings
x=0                        #integer for storing the duty cycle value
IO.setmode (IO.BCM)       #we are programming the GPIO by BCM pin numbers. (PIN35
as,,GPIO19")
IO.setup(13,IO.OUT)        # initialize GPIO13 as an output.
IO.setup(19,IO.IN)         # initialize GPIO19 as an input.
IO.setup(26,IO.IN)        # initialize GPIO26 as an input.
p = IO.PWM(13,100)        #GPIO13 as PWM output, with 100Hz frequency
p.start(0)                 #generate PWM signal with 0% duty cycle
while 1:                  #execute loop forever
    p.ChangeDutyCycle(x)   #change duty cycle for changing the brightness of LED.
    if(IO.input(26) == False): #if button1 is pressed
        if(x<50):
            x=x+1          #increment x by one if x<50
            time.sleep(0.2) #sleep for 200ms
        if(IO.input(19) == False): #if button2 is pressed
            if(x>0):
                x=x-1       #decrement x by one if x>0
                time.sleep(0.2) #sleep for 200ms
```

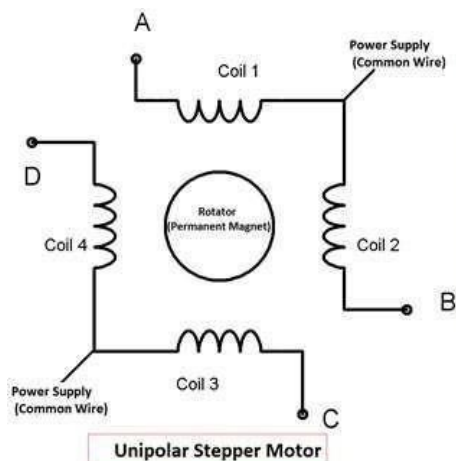


Stepper Motor Control with Raspberry Pi

Raspberry Pi is an ARM architecture processor based board designed for electronic engineers and hobbyists. The PI is one of most trusted project development platforms out there now. With higher processor speed and 1 GB RAM, the PI can be used for many high profile projects like Image processing and **Internet of Things**.

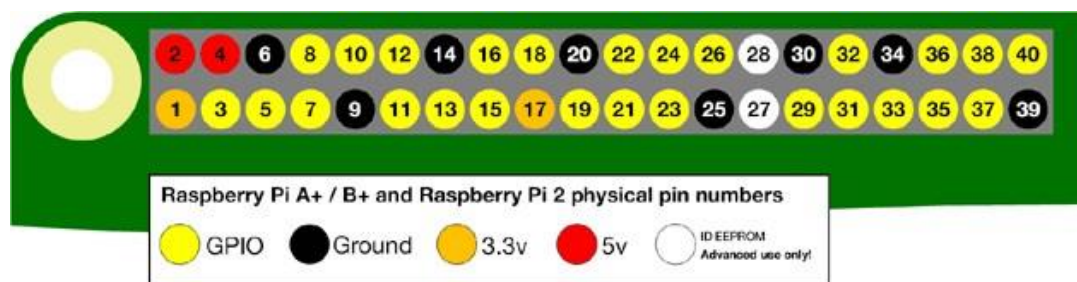
For doing any of high profile projects, one need to understand the basic functions of PI. We will be covering all the **basic functionalities of Raspberry Pi** in these tutorials. In each tutorial we will discuss one of functions of PI. By the end of this **Raspberry Pi Tutorial Series**, you will be able to do high profile projects by yourself.

In this tutorial, we will **Control the Speed of a Stepper Motor using Raspberry Pi**. In Stepper Motor, as the name itself says, the rotation of shaft is in Step form. There are different types of Stepper Motor; in here we will be using the most popular one thatis **Unipolar Stepper Motor**. Unlike DC motor, we can rotate stepper motor to any particular angle by giving it proper instructions.



To rotate this Four Stage Stepper Motor, we will deliver power pulses by using **Stepper Motor Driver Circuit**. The driver circuit takes logic triggers from PI. If we control the logic triggers, we control the power pulses and hence the speed of stepper motor.

There are **40 GPIO output pins in Raspberry Pi 2**. But out of 40, only 26 GPIO pins (GPIO2 to GPIO27) can be programmed. Some of these pins perform some special functions. With special GPIO put aside, we have only 17 GPIO remaining. Each of these 17 GPIO pin can deliver a maximum of **15mA** current. And the sum of currents from all GPIO Pins cannot exceed **50mA**. To know more about GPIO pins, go through: **LED Blinking with Raspberry Pi**



There are **+5V (Pin 2 & 4) and +3.3V (Pin 1 & 17) power output pins** on the board for connecting other modules and sensors. These power rails cannot be used to drive the Stepper Motor, because we need more power to rotate it. So we have to deliver the power to Stepper Motor from another power source. My stepper motor has a voltage rating of 9V so I am using a 9v battery as my second power source. Search your stepper motor model number to know voltage rating. Depending on the rating choose the secondary source appropriately.

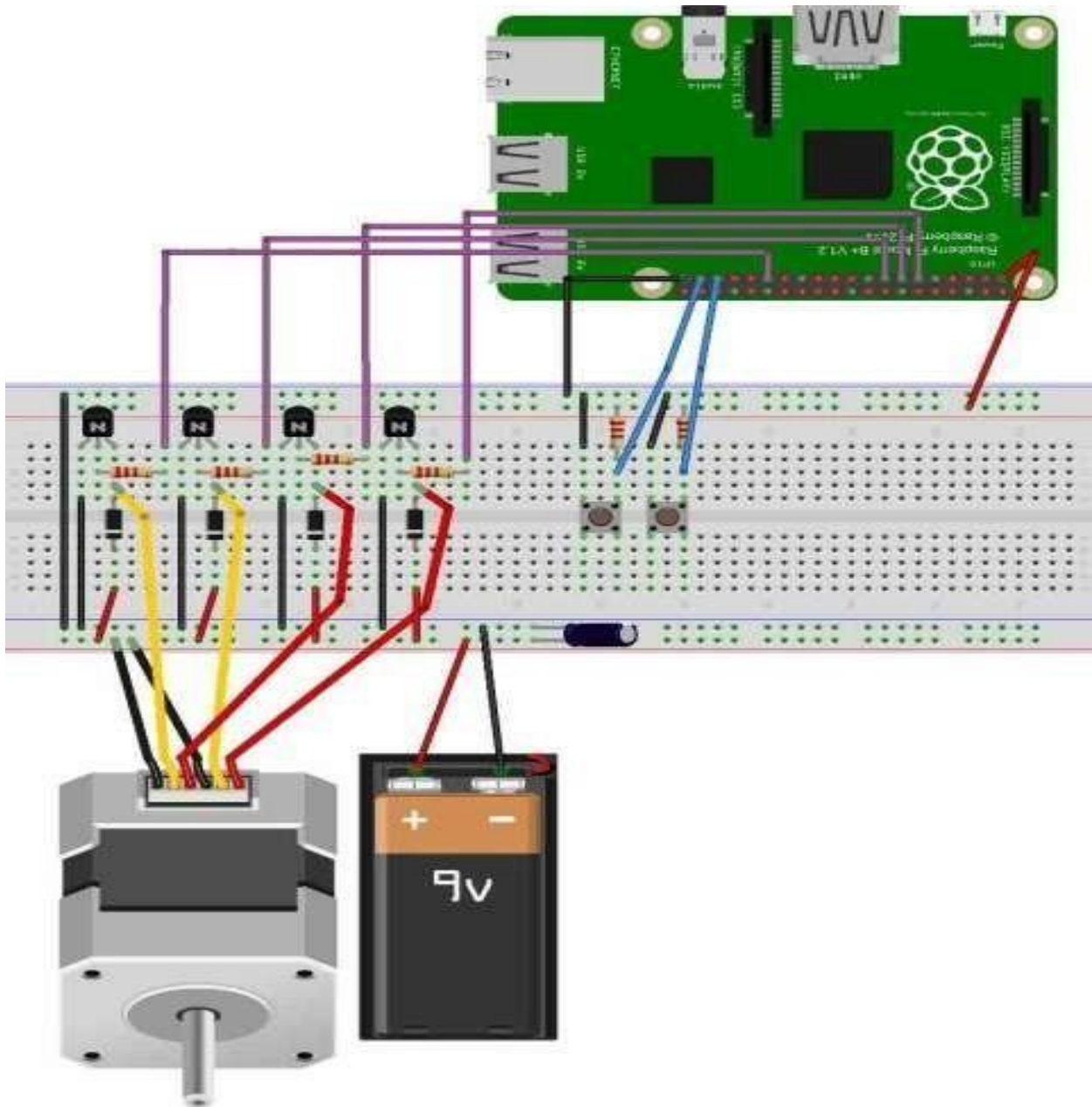
As stated earlier, we need a driver circuit to drive the Stepper Motor. We will also be designing a Simple Transistor Driver Circuit here.

Components Required:

Here we are using **Raspberry Pi 2 Model B with Raspbian Jessie OS**. All the basic Hardware and Software requirements are previously discussed, you can look it up in the **Raspberry Pi Introduction**, other than that we need:

1. Connecting pins
2. 220Ω or $1K\Omega$ resistor (3)
3. Stepper Motor
4. Buttons (2)
5. 2N2222 Transistor (4)
6. 1N4007 Diode (4)
7. Capacitor- 1000uF, Bread Board

Circuit Explanation:



Stepper motor use **200 steps to complete 360 degree** rotation, means its rotate **1.8 degree per step**. As we are driving a Four Stage Stepper Motor, so we need to give four pulses to complete single logic cycle. Each step of this motor completes 1.8 degree of rotation, so in order to complete a cycle we need 200 pulses. So $200/4 = 50$ logic cycles needed to complete a single rotation. Check this to know more about **Steppers Motors and its Driving Modes**.

We will be driving each of these four coils by a **NPN transistor (2N2222)**, this NPN transistor takes the logic pulse from PI and drives the corresponding coil. Four transistors are taking four logics from PI to drive four stages of stepper motor.

The transistor driver circuit is a tricky setup; here we should pay attention that wrongly connecting the transistor might load the board heavily and damage it. Check this to properly understand the **Stepper Motor Driver Circuit**.

The motor is an induction and so while switching the motor, we experience inductive spiking. This spiking will heat up the transistor heavily, so we will be **using Diode (1N4007)** to provide protection to transistor **against Inductive Spiking**.

In order **to reduce the voltage fluctuations**, we will be connecting a **1000uF capacitor** across the power supply as shown in the Circuit Diagram.

Working Explanation:

Once everything is connected as per the circuit diagram, we can turn ON the PI to write the program in PYHTON.

We will talk about few commands which we are going to use in **PYHTON program**, We are going to import GPIO file from library, below function enables us to program

GPIO pins of PI. We are also renaming “GPIO” to “IO”, so in the program whenever we want to refer to GPIO pins we will use the word „IO“.

```
import RPi.GPIO as IO
```

Sometimes, when the GPIO pins, which we are trying to use, might be doing some other functions. In that case, we will receive warnings while executing the program. Below command tells the PI to ignore the warnings and proceed with the program.

```
IO.setwarnings(False)
```

We can refer the GPIO pins of PI, either by pin number on board or by their function number. Like „PIN 35“ on the board is „GPIO19“. So we tell here either we are going to represent the pin here by „35“ or „19“.

```
IO.setmode (IO.BCM)
```

We are setting four of GPIO pins as output for driving four coils of stepper motor.

```
IO.setup(5,IO.OUT)
IO.setup(17,IO.OUT)
IO.setup(27,IO.OUT)
IO.setup(22,IO.OUT)
```

We are setting GPIO26 and GPIO19 as input pins. We will detect button press by these pins.

```
IO.setup(19,IO.IN)
IO.setup(26,IO.IN)
```

In case the Condition in the braces is true, the statements inside the loop will be executed once. So if the GPIO pin 26 goes low, then the statements inside the IF loop will be executed once. If the GPIO pin 26 does not goes low, then the statements inside the IF loop will not be executed.

```
if(IO.input(26) == False):
```

```
for x in range (100):
```

While 1: is used for infinity loop. With this command the statements inside this loop will be executed continuously.

We have all the commands needed to achieve the **Speed Control of Stepper Motor** with this.

After writing the program and executing it, all there is left is operating the control. We have two buttons connected to PI. One for increments the delay between the four pulses and other for decrements the delay between the four pulses. The delay itself speaks of speed; if the **delay is higher** the motor takes brakes between each step and so **rotation is slow**. If the **delay is near zero, then the motor rotates at maximum speed**.

Here it should be remember that, there should be some delay between the pulses. After giving a pulse, stepper motor takes few milliseconds of time to reach its final stage. If there is no delay given between the pulses, the stepper motor will not move at all. Normally 50ms delay is fine between the pulses. For more accurate information, look into the data sheet.

So with two buttons we can control the delay, which in turns control the speed of the stepper motor.

Program:

```
import RPi.GPIO as IO # we are calling for header file which helps us use GPIO's of PI# we are
#calling for time to provide delays in program
time IO.setwarnings(False) # do not show any warnings

x=1      # integer for storing the delay multipleIO.setmode (IO.BCM)
IO.setup(5,IO.OUT)          # initialize GPIO5 as an output.IO.setup(17,IO.OUT)
IO.setup(27,IO.OUT)IO.setup(22,IO.OUT)
IO.setup(19,IO.IN)          # initialize GPIO19 as an input.IO.setup(26,IO.IN)
while 1:    # execute loop foreverIO.output(5,1) # Step1 go high IO.output(22,0)
for y in range(x):          # sleep for x*100msectime.sleep(0.01)
    IO.output(17,1)          # step2 go highIO.output(5,0)
    for y in range(x):
        time.sleep(0.01)     # sleep for x*100msecIO.output(27,1)      #step 3 go high
    IO.output(17,0)
    for y in range(x):
        time.sleep(0.01)     # sleep for x*100msecIO.output(22,1)      #step 4 go high
    IO.output(27,0)
    for y in range(x):
        time.sleep(0.01)     # sleep for x*100msecif(IO.input(26) == False): #if
        button1 is pressed
    if(x<100):
        x=x+1                #increment x by one if x<100
        time.sleep(0.5)       #sleep for 500ms if(IO.input(19) == False):    #if
        button2 is pressed
    if(x>1):
        x=x-1                #decrement x by one if x>1
        time.sleep(0.5)       #sleep for 500ms
```



Interfacing DS18B20 Temperature Sensor with Raspberry Pi

Raspberry Pi is known for its computational power and its vast application in the field of [IoT](#), Home Automation etc. However for any electronic system to interact with the real world and get information about it, the system has to use sensors. There are many types of sensors used for this process and the required sensor is selected based on the parameter to be measured and its application. In this tutorial we learn to **interface a temperature sensor DS18B20 with the Raspberry Pi**.

The **DS18B20** is widely used temperature sensor, mainly at places where harsh operating environments are involved like chemical industries, mine plants etc. This article will tell about the sensor and how it outstands other temperature sensor and finally interface it with Raspberry Pi and view the temperature value on the 16x2 LCD.

Components Required

1. DS18B20 Temperature Sensor
2. Raspberry Pi
3. 16*2 LCD display
4. 10k trim pot
5. 10k Pull up resistor
6. Breadboard
7. Connecting wires

Introduction to DS18B20 Temperature Sensor

The DS18B20 is a three terminal temperature sensor available in the TO-92 (transistor type) package. It is very easy to use and requires only one external component to start working. Also it requires only one GPIO pin from the MCU/MPU to communicate with it. A typical DS18B20 temperature sensor with its pin name is shown below.



This sensor is also available as a waterproof version in which the sensor is covered by a cylindrical metal tube. In this tutorial we will be using the normal transistor type sensor that is shown above. The **DS18B20 is a 1-wire programmable temperature sensor** meaning it requires only the data pin to send the information to the microcontroller or microprocessor boards like the Raspberry Pi. Each sensor has a unique address of 64-bit for it so it is also possible to have multiple sensors connected to the same MCU/MPU since each sensor can be addressed individually on the same data bus. The specification of the sensor is shown below.

- Operating voltage: 3-5V
- Measuring Range: -55°C to $+125^{\circ}\text{C}$
- Accuracy: $\pm 0.5^{\circ}\text{C}$
- Resolution: 9-bit to 12-bit

Now that we know enough of the sensor, let us start interfacing it with Raspberry Pi.

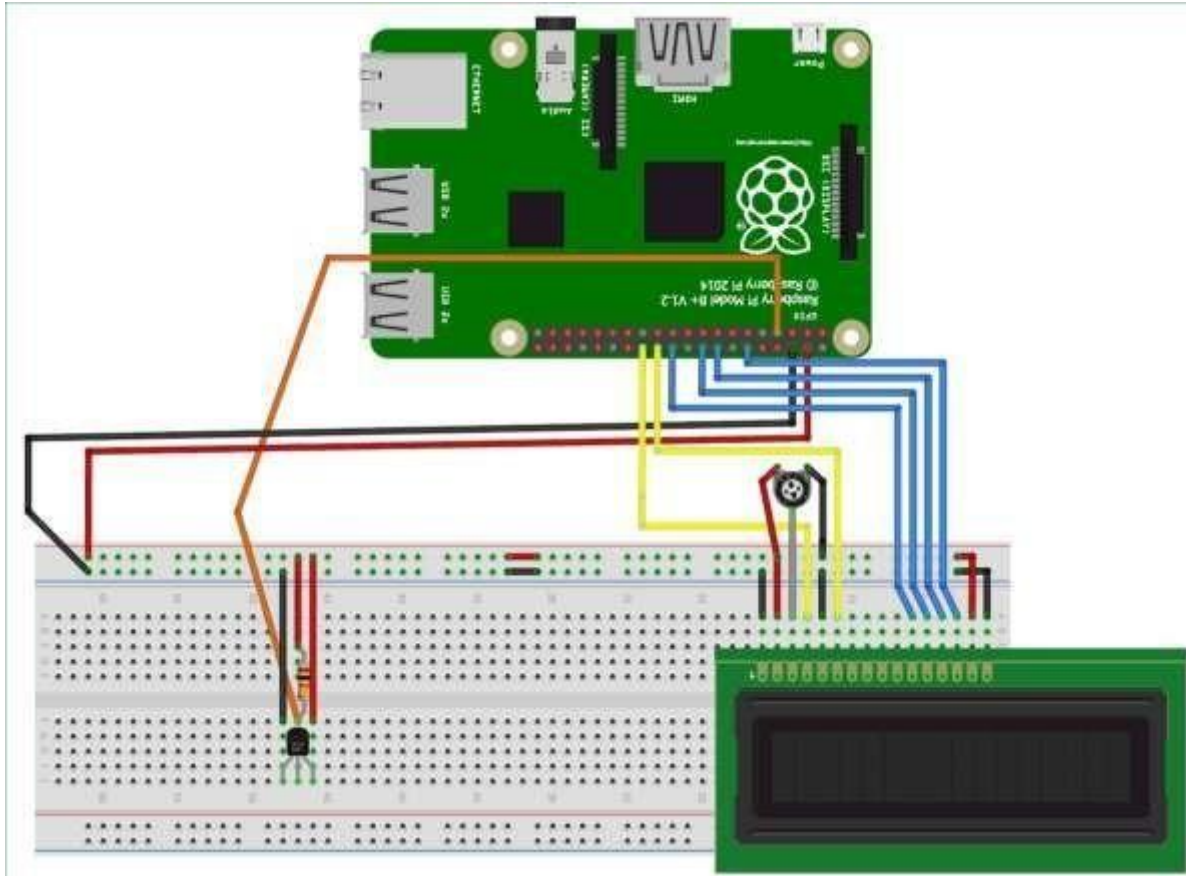
Pre-Requisites

It is assumed that your Raspberry Pi is already flashed with an operating system and is able to connect to the internet. If not, follow the **Getting started with Raspberry Pi** tutorial before proceeding. Here we are using **Raspbian Jessie installed Raspberry Pi 3**.

It is also assumed that you have access to your pi either through terminal windows or through other application through which you can write and execute python programs and use the terminal window.

Circuit Diagram

As we told earlier in this tutorial we will **interface the DS18B20 sensor with Pi** and display the value of temperature on a 16*2 LCD screen. So the sensor and the LCD should be connected with Raspberry Pi as show below.

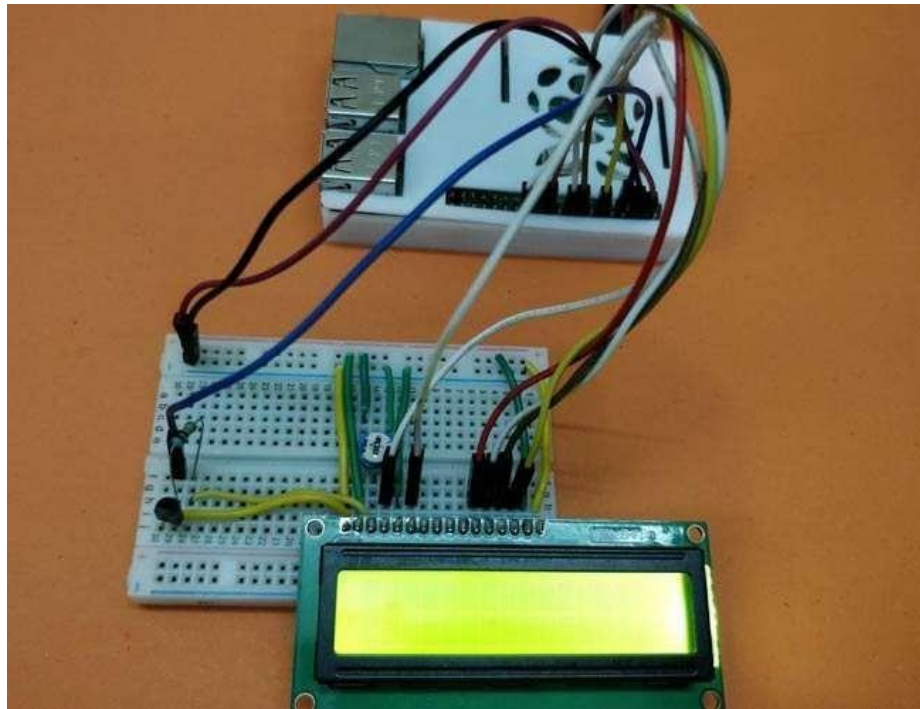


Follow the circuit diagram and make the connection accordingly. Both the LCD and the **DS18B20** sensor works with the help of +5V which is provided by the 5V pin on the Raspberry pi. The **LCD** is made to work in 4-bit mode with Raspberry pi, the GPIO pins 18,23,24 and 25 is used for the data line and the GPIO pins 7 and 8 is used for the control lines. A **potentiometer** is also used to control the contrast level of the LCD. The DS18B20's data line is connected to GPIO pin 4. Also note that a 10K resistor must be used pull the data line high as show in the circuit diagram.

You can either follow the circuit diagram above and make the connections or use the pin table to follow up with the GPIO pin numbers.



I have built the circuit on a breadboard using the single strand wires and male to female wires to make the connections. As you can see the sensor requires only one wire to interface and hence occupies less space and pins. My hardware looks like this below when all the connections are made. Now it time to power up the pi and start programming.



Installing the Adafruit LCD library on Raspberry Pi

The value of the temperature will be displayed on a 16*2 LCD display. Adafruit provides us a library to easily operate this LCD in 4-bit mode, so let us add it to our Raspberry Pi by opening the terminal window Pi and following the below steps.

Step 1: Install git on your Raspberry Pi by using the below line. **Git** allows you to clone any project files on Github and use it on your Raspberry pi. Our library is on Github so we have to install git to download that library into pi.

```
apt-get install git
```

Step 2: The following line links to the GitHub page where the library is present just execute the line to clone the project file on Pi home directory

```
git clone git://github.com/adafruit/Adafruit_Python_CharLCD
```

Step 3: Use the below command to change directory line, to get into the project file that we just downloaded. The command line is given below

```
cd Adafruit_Python_CharLCD
```

Step 4: Inside the directory there will be a file called *setup.py*, we have to install it, to install the library. Use the following code to install the library

```
sudo python setup.py install
```

That is it **the library should have been installed successfully**. Now similarly let's proceed with installing the DHT library which is also from Adafruit.

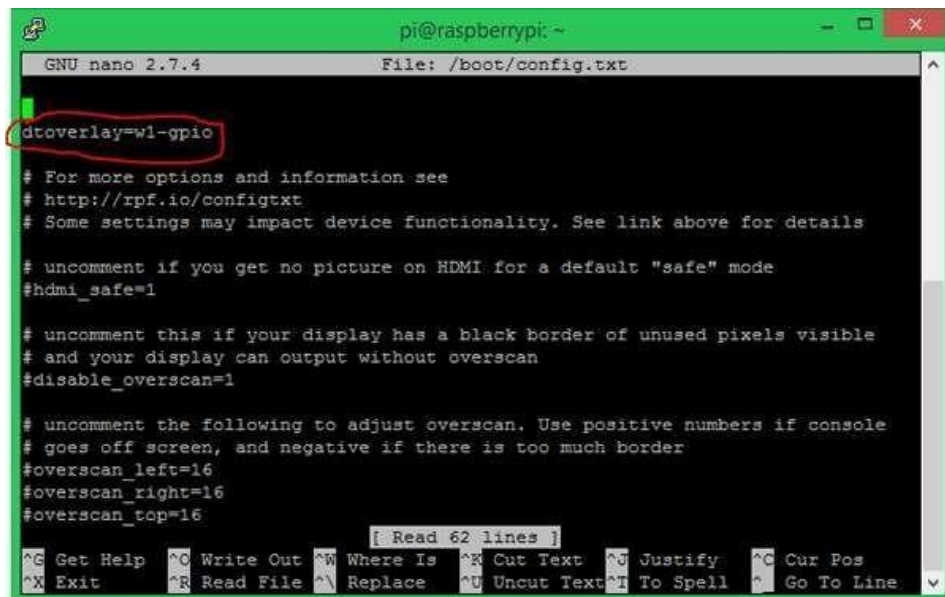
Enabling One-Wire Interface in Pi

Since the DS18B20 sensor communicates through One-Wire method, we have to enable the one wire communication on Pi by following the below steps.

Step 1:- Open the Commands prompt and use the below command to open the config file

```
sudo nano /boot/config.txt
```

Step 2:- Inside the *config* file add the line "*dtoverlay=w1-gpio*" (encircled in below image) and save the file as shown below



```
pi@raspberrypi: ~
GNU nano 2.7.4 File: /boot/config.txt
dtoverlay=w1-gpio
# For more options and information see
# http://rpf.io/config.txt
# Some settings may impact device functionality. See link above for details
# uncomment if you get no picture on HDMI for a default "safe" mode
#hdmi_safe=1
# uncomment this if your display has a black border of unused pixels visible
# and your display can output without overscan
#disable_overscan=1
# uncomment the following to adjust overscan. Use positive numbers if console
# goes off screen, and negative if there is too much border
#overscan_left=16
#overscan_right=16
#overscan_top=16
[ Read 62 lines ]
^G Get Help ^O Write Out ^W Where Is ^X Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

Step 3:- Use Ctrl+X to exit the file and save it by pressing "Y" and then Enter key. Finally **restart the Pi** by using the command

```
sudo reboot
```

```
sudo modprobe w1-gpio
sudo modprobe w1-therm.
cd /sys/bus/w1/devices
ls
```

Step 4:- Once rebooted, open the terminal again and enter the following commands.

Your terminal windows will display something like this



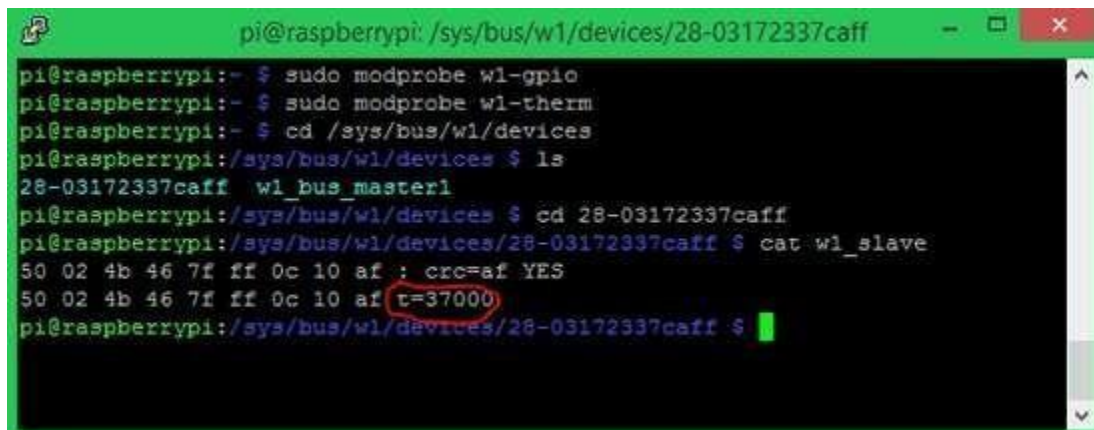
```
pi@raspberrypi: /sys/bus/w1/devices
pi@raspberrypi:~$ sudo modprobe w1-gpio
pi@raspberrypi:~$ sudo modprobe w1-therm
pi@raspberrypi:~$ cd /sys/bus/w1/devices
pi@raspberrypi:/sys/bus/w1/devices$ ls
28-03172337caff  w1_bus_master1
pi@raspberrypi:/sys/bus/w1/devices$
```

Step 5:- At the end of step 4 when you enter *ls*, your pi will display a unique number this number will be different for each user, based on the sensor, but will always start with 28-. In my case the number is 28-03172337caff.

Step 6:- Now we can check if the sensor is working by entering the following commands

```
cd 28-XXXXXXXXXXXX [use the name of your directory or use Tab key for auto complete)
cat w1_slave
```

These two commands will read the data from the sensor and display it on the terminal as show below. The value of temperature is encircled with red in the below picture. In my case the value of temperature is 37°C.



```
pi@raspberrypi: /sys/bus/w1/devices/28-03172337caff
pi@raspberrypi:~$ sudo modprobe w1-gpio
pi@raspberrypi:~$ sudo modprobe w1-therm
pi@raspberrypi:~$ cd /sys/bus/w1/devices
pi@raspberrypi:/sys/bus/w1/devices$ ls
28-03172337caff  w1_bus_master1
pi@raspberrypi:/sys/bus/w1/devices$ cd 28-03172337caff
pi@raspberrypi:/sys/bus/w1/devices/28-03172337caff$ cat w1_slave
50 02 4b 46 7f ff 0c 10 af : crc=af YES
50 02 4b 46 7f ff 0c 10 af t=37000
pi@raspberrypi:/sys/bus/w1/devices/28-03172337caff$
```

Programming your Raspberry Pi for DS18B20 Sensor

Now we have our Pi ready to be programmed for LCD and to use One-wire protocol. So we can write our final program to read the value of temperature from the DS18B20 sensor and display it on the LCD screen. The **complete python program** to do the same is given at the end of this page. However below I have split the code into small meaningful snippets to explain them.

As always we begin the program, by **importing the header files** required by the program. Here we import *time* to deal with *delay* function, the *LCD* header to use *LCD* with *Pi*. The *os* header is used to handling files in the *OS*.

```
import time #import time for creating delay
import Adafruit_CharLCD as LCD #Import LCD library
import os #Import for file handling
import glob #Import for global
```

Next we have to mention the **LCD pins which are connected to Raspberry Pi Pins**. Use the GPIO pin chart provided above to know the pin numbers of the respective GPIO pins. Once we have declared, to which pins of *PI* the *LCD* is connected to, we can specify the number of rows and columns and finally initialize it by using the below lines of code.

```
lcd_rs    = 7 #RS of LCD is connected to GPIO 7 on PI
lcd_en    = 8 #EN of LCD is connected to GPIO 8 on PI
lcd_d4    = 25 #D4 of LCD is connected to GPIO 25 on PI
lcd_d5    = 24 #D5 of LCD is connected to GPIO 24 on PI
lcd_d6    = 23 #D6 of LCD is connected to GPIO 23 on PI
lcd_d7    = 18 #D7 of LCD is connected to GPIO 18 on PI
lcd_backlight = 0 #LED is not connected so we assign to 0

lcd_columns = 16 #for 16*2 LCD
lcd_rows    = 2 #for 16*2 LCD

lcd = LCD.Adafruit_CharLCD(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7,
    lcd_columns, lcd_rows, lcd_backlight) #Send all the pin details to library
```

After initializing the *LCD* we **print a sample text message on the LCD**. The character „\n“ is used to mention new line. After displaying the intro we introduce a delay of 2 seconds for the user to read the intro message.

```
lcd.message('DS18B20 with Pi \n -CircuitDigest') #Give a intro message
time.sleep(2) #wait for 2 secs
```


Now, if you could recollect the step 4 of enabling one wire interface with Pi. We have to repeat the same line of code, so we use the *os.system* function to execute the same lines. Then we specify the file location from where the value of temperature has to be read. The *device_folder* variable points to the folder that starts with „28-“ since we do not know the exact name of the folder we use the * symbol to open whatever that starts with 28. Finally inside that folder we use another variable called *device_file* which actually points to the file which has the value of temperature inside it.

Then we use function named *get_temp* inside which we define the procedure of reading the temperature from the file that we just linked in the above step. As we checked with the terminal earlier the file will have the value of temperature inside it but it will be in the following format



```
50 02 4b 46 7f ff 0c 10 af : crc=af YES
50 02 4b 46 7f ff 0c 10 af t=37000
```

From this we only need the value of 37000, which is the **value of temperature**. Here the actual value of temperature is 37.00°C. So from this format of text we have to **trim all the useless data and get the integer value 37000 and finally divide it by 1000** to get the actual data. The function shown below does exactly the same

```
def get_temp(): #Function to read the value of Temperature
    file = open(device_file, 'r') #open the file
    lines = file.readlines() #read the lines in the file
    file.close() #close the file
    trimmed_data = lines[1].find('t=') #find the "t=" in the line
    if trimmed_data != -1:
        temp_string = lines[1][trimmed_data+2:] #trim the string only to the temperature value
        temp_c = float(temp_string) / 1000.0 #divide the value of 1000 to get actual value
        return temp_c #return the value to print on LCD
```

The variable *lines* is used to read the lines inside the file. Then these lines are compared searched for the letter “t=” and the value after that letter is saved in the variable *temp_string*. Finally to get the value of temperature we use the variable *temp_c* in which we divide the string value by 1000. In the end return the *temp_c* variable as a result of the function.

Inside the infinite *while* loop, we only have to call the above defined function to **get the value of temperature and display it in the LCD screen**. We also clear the LCD for every 1 sec to display the updated value.

```
while 1: #Infinite Loop
```

```
    lcd.clear() #Clear the LCD screen
```

```
    lcd.message ('Temp = %.1f C' % get_temp()) # Display the value of temperature
```

```
    time.sleep(1) #Wait for 1 sec then update the values
```

Output / Working

As always the **complete python code** is given at the end of the page, use the code and compile it on your Raspberry Pi. Make the connection as shown in the circuit diagram and before executing the program make sure you have followed the above steps to install LCD header files and enable one-wire communication on pi. Once that is done just execute the program, if everything is working as expected you should be able to notice the intro text. If not adjust the contrast potentiometer until you see something. The final result will look something like this below.



Hope you understood the project and had no problem building it. If otherwise state your problem in the comment section or use the forum for more technical help. This is just an interfacing project, but once this is done you can think ahead by working on a Raspberry Pi weather station, temperature E-mail notifier and much more.

The complete working of the project is also show in the **video** below where you cansee the value of temperature being updated in real time.

Code

```
#Program to read the values of Temp from the DS18B20 sensor and display them on the LCD
import time #import time for creating delay
import Adafruit_CharLCD as LCD #Import LCD libraryimport os #Import for file handling
import glob #Import for global

lcd_rs          = 7 #RS of LCD is connected to GPIO 7 on PI lcd_en          = 8 #EN of
LCD is connected to GPIO 8 on PI lcd_d4          = 25 #D4 of LCD is connected to GPIO 25
on PIlcd_d5          = 24 #D5 of LCD is connected to GPIO 24 on PIlcd_d6          = 23 #D6 of
LCD is connected to GPIO 23 on PIlcd_d7          = 18 #D7 of LCD is connected to GPIO 18
on PIlcd_backlight = 0 #LED is not connected so we assign to 0 lcd_columns = 16 #for 16*2 LCD
lcd_rows          = 2 #for 16*2 LCD

lcd = LCD.Adafruit_CharLCD(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, lcd_columns,
lcd_rows, lcd_backlight) #Send all the pin details to library
lcd.message('DS18B20 with Pi \n -CircuitDigest') #Give a intro messagetime.sleep(2) #wait for 2
secs

os.system('modprobe w1-gpio') os.system('modprobe w1-therm')base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]device_file = device_folder + '/w1_slave'

def get_temp(): #Functon to read the value of Temperaturefile = open(device_file, 'r') #opent the
file

lines = file.readlines() #read the lines in the filefile.close() #close the file

trimmed_data = lines[1].find('t=') #find the "t=" in the lineif trimmed_data != -1:
temp_string = lines[1][trimmed_data+2:] #trim the strig only to the temoerature value
temp_c = float(temp_string) / 1000.0 #divide the value of 1000 to get actual valuereturn temp_c
#return the value to prnt on LCD

while 1: #Infinite Loop
lcd.clear() #Clear the LCD screen

lcd.message ('Temp = %.1f C' % get_temp()) # Display the value of temperaturetime.sleep(1) #Wait
for 1 sec then update the values
```