SSG EMBEDDED SOLUTIONS

SSG

info@ssges.co.in

DEVLOPMENT KIT RASPBERRY Pi 4 DOCUMENTATION

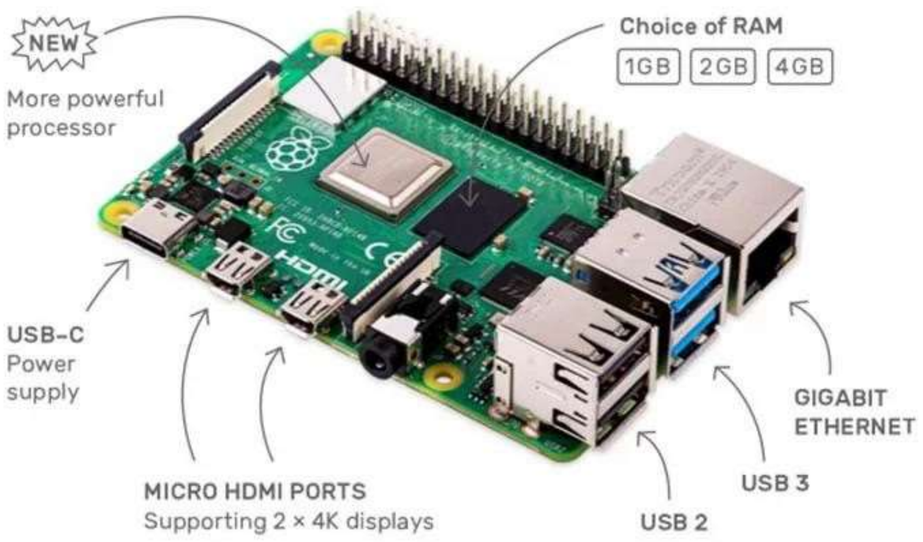
List of Content

Introduction

Raspberry Pi is a small single board computer. By connecting peripherals like Keyboard, mouse, display to the Raspberry Pi, it will act as a mini personal computer.

Raspberry Pi is popularly used for real time Image/Video Processing, IoT based applications and Robotics applications.

Raspberry Pi is slower than laptop or desktop but is still a computer which can provide all the expected features or abilities, at a low power consumption.



Raspberry Pi Foundation officially provides Debian based Raspbian OS. Also, they provide NOOBS OS for Raspberry Pi. We can install several Third-Party versions of OS like Ubuntu, Archlinux, RISC OS, Windows 10 IOT Core, etc.

Raspbian OS is official Operating System available for free to use. This OS is efficiently optimized to use with Raspberry Pi. Raspbian have GUI which includes tools for Browsing, Python programming, office, games, etc.

We should use SD card (minimum 8 GB recommended) to store the OS (operating System).

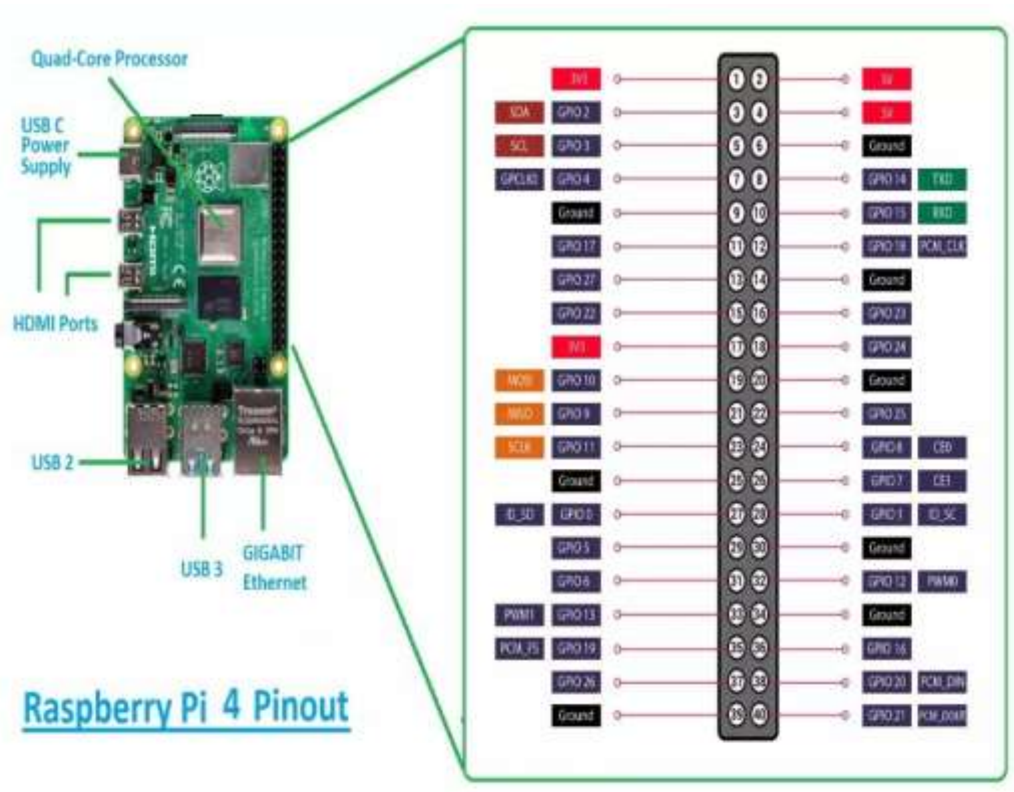
Raspberry Pi is more than computer as it provides access to the on-chip hardware i.e. GPIOs for developing an application. By accessing GPIO, we can connect devices like LED, motors, sensors, etc and can control them too.

It has ARM based Broadcom Processor SoC along with on-chip GPU (Graphics Processing Unit).

The CPU speed of Raspberry Pi varies from 700 MHz to 1.2 GHz. Also, it has on-board SDRAM that ranges from 256 MB to 1 GB.

Raspberry Pi also provides on-chip SPI, I2C, I2S and UART modules.

Pin Configuration


Quad-Core Processor
USB C Power Supply
HDMI Ports
USB 2
USB 3
GIGABIT Ethernet
Raspberry Pi 4 Pinout
3V3 1 2 5V
SDA GPIO 2 3 4 5V
SCL GPIO 3 5 6 Ground
GPCLK0 GPIO 4 7 8 GPIO 14 TXD
Ground 9 10 GPIO 15 RXD
GPIO 17 11 12 GPIO 18 PCM_CLK
GPIO 27 13 14 Ground
GPIO 22 15 16 GPIO 23
3V3 17 18 GPIO 24
MOSI GPIO 10 19 20 Ground
MISO GPIO 9 21 22 GPIO 25
SCLK GPIO 11 23 24 GPIO 8 CE0
Ground 25 26 GPIO 7 CE1
ID_SD GPIO 0 27 28 GPIO 1 ID_SC
GPIO 5 29 30 Ground
GPIO 6 31 32 GPIO 12 PWM0
PWM1 GPIO 13 33 34 Ground
PCM_FS GPIO 19 35 36 GPIO 16
GPIO 26 37 38 GPIO 20 PCM_DIN
Ground 39 40 GPIO 21 PCM_DOUT

Features

- 512 MB SDRAM memory
- Broadcom BCM2835 SoC full high definition multimedia processor
- Dual Core Video Core IV Multimedia coprocessor
- Single 2.0 USB connector
- HDMI (rev 1.3 and 1.4) Composite RCA (PAL & NTSC) Video Out
- 3.5 MM Jack, HDMI Audio Out
- MMC, SD, SDIO Card slot on board storage
- Linux Operating system
- Dimensions are 8.6cm*5.4cm*1.7cm
- On board 10/100 Ethernet RJ45 jack

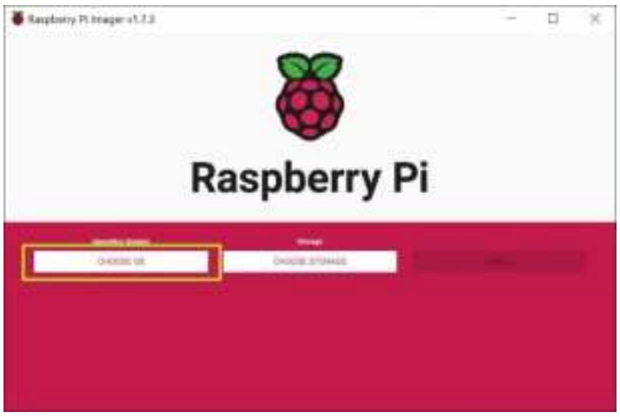
Applications

The raspberry pi boards are used in many applications like Media streamer, Arcade machine, Tablet computer, Home automation, Carputer, Internet radio, Controlling robots, Cosmic Computer, Hunting for meteorites, Coffee and also in raspberry pi based projects.

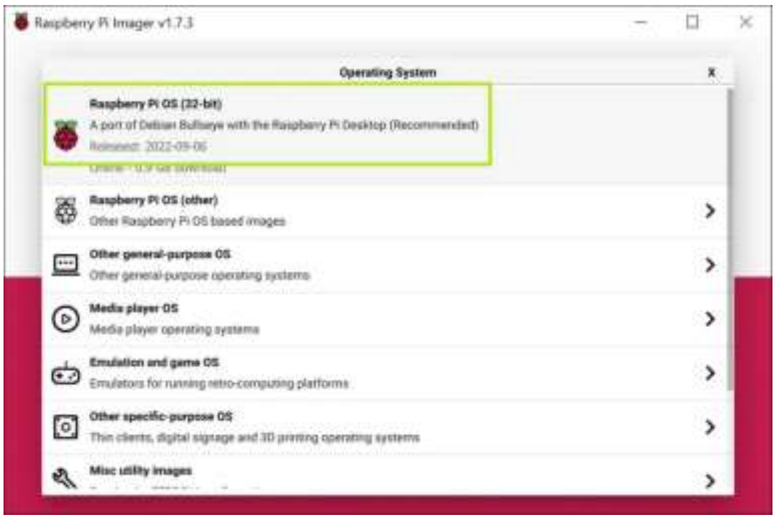
Downloading and Installing Raspberry Pi OS

Once you have all the components you need, use the following steps to create the boot disk you will need to set up your Raspberry Pi. These steps should work on a using a Windows, Mac or Linux-based PC (we tried this on Windows, but it should be the same on all three).

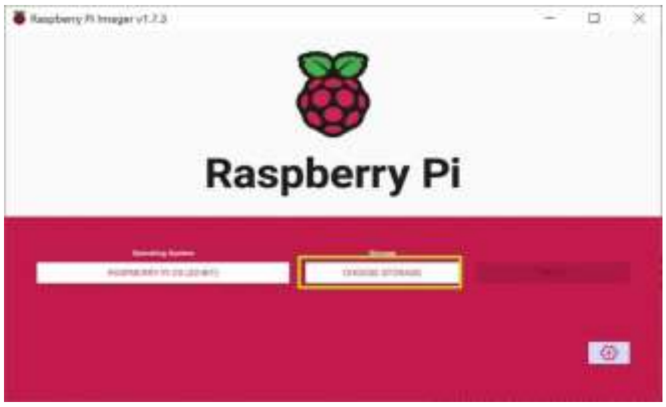
1. **Insert a microSD card** / **reader** into your computer.
2. **Download and install the official Raspberry Pi Imager.** Available for Windows, macOS or Linux, this app will both download and install the latest Raspberry Pi OS. There are other ways to do this, namely by downloading a Raspberry Pi OS image file and then using a third-party app to "burn it," but the Imager makes it easier.
3. **Click Choose OS.**



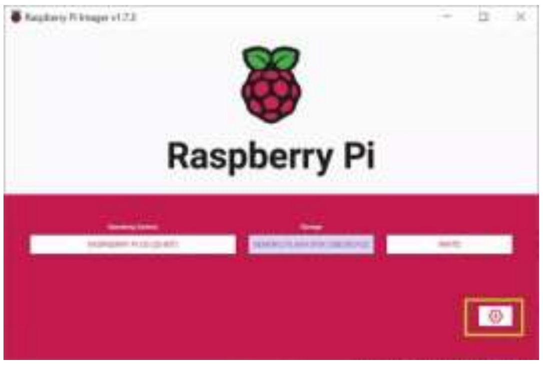
4. **Select Raspberry Pi OS (32-bit)** from the OS menu (there are other choices, but for most uses, 32-bit is the best).


Raspberry Pi Imager v1.7.3
Operating System
Raspberry Pi OS (32-bit)
A port of Debian Bullseye with the Raspberry Pi Desktop (Recommended)
Released: 2022-09-06
Raspberry Pi OS (other)
Other Raspberry Pi OS based images
Other general-purpose OS
Other general-purpose operating systems
Media player OS
Media player operating systems
Emulation and game OS
Emulators for running retro-computing platforms
Other specific-purpose OS
Thin clients, digital signage and 3D printing operating systems
Misc utility images

4. **Click Choose storage** and **pick the SD card** you're using.

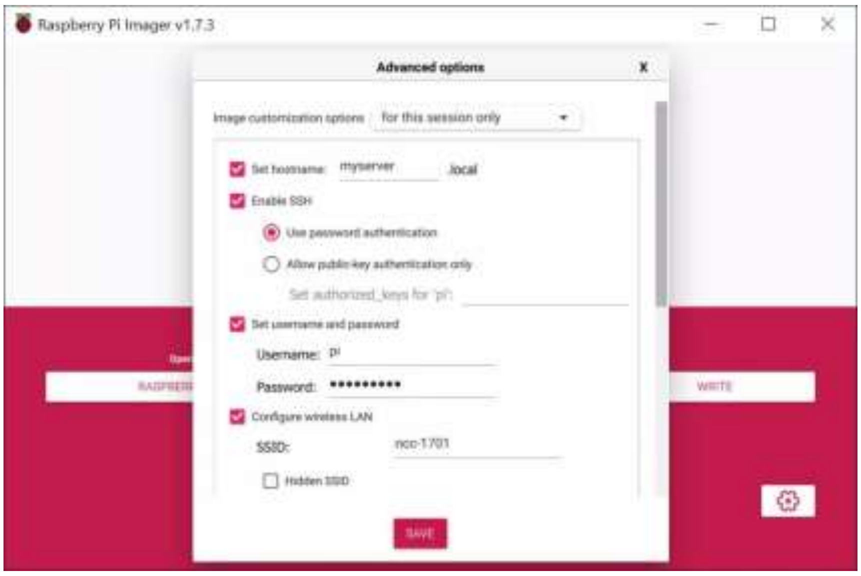


5. **Click the settings button** or hit CTRL + SHIFT + X to enter settings.



6. **Fill in settings fields** as follows and then **hit Save**. All of these fields are technically optional, but highly recommended so that can get your Raspberry Pi set up and online as soon as you boot it. If you don't set a username and password here, you'll have to go through a setup wizard that asks you to create them on first boot.

- **Set hostname**: the name of your Pi. It could be "raspberrypi" or anything you like.
- **Enable SSH**: Allow SSH connections to the Pi. Recommended.
- **Use password authentication / public key:** method of logging in via SSH
- **Set username and password:** Pick the username and password you'll use for the Pi
- **Configure wireless LAN:** set the SSID and password of Wi-FI network
- **Wireless LAN country:** If you're setting up Wi-Fi, you must choose this.
- **Set locale settings:** Configure keyboard layout and timezone (probably chosen correctly by default)

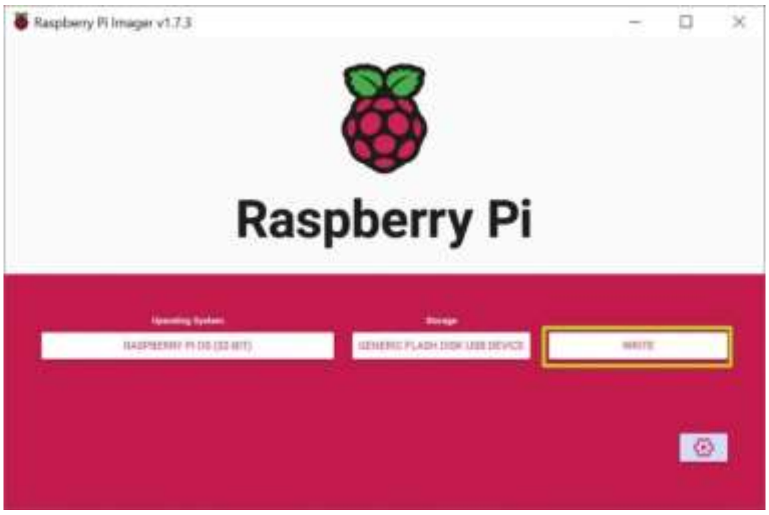

Raspberry Pi Imager v1.7.3

Advanced options

Image customization options: for this session only

[✓] Set hostname: myserver .local
[✓] Enable SSH
(●) Use password authentication
( ) Allow public-key authentication only
Set authorized_keys for 'pi':
[✓] Set username and password
Username: pi
Password: **********
[✓] Configure wireless LAN
SSID: ncc-1701
[ ] Hidden SSID

7. **Click Write.** The app will now take a few minutes to download the OS and write to your card.


Raspberry Pi Imager v1.7.3

Booting Your Raspberry Pi for the First Time

After you're done writing the Raspberry Pi OS to a microSD card, it's time for the moment of truth.

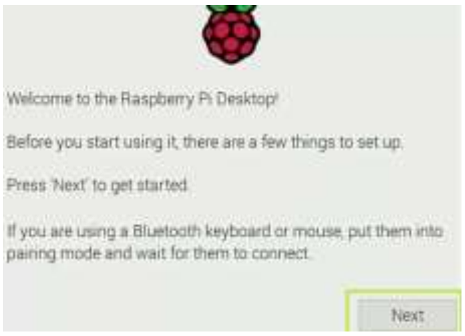
1. **Insert the microSD card** into the Raspberry Pi.
2. **Connect the Raspberry Pi** to a monitor, keyboard and mouse.
3. **Connect an Ethernet cable** if you plan to use wired Internet.
4. **Plug the Pi in** to power it on.

If you had used the Raspberry Pi Imager settings to create a username and password, you'll be able to go straight into the desktop environment, but if not, you will get a setup wizard.

Using the Raspberry Pi First-Time Setup WIzard

If you chose a username and password in Raspberry Pi Imager settings, before writing the microSD card, you will get the desktop on first boot. But, if you did not, you'll be prompted to create a username and password and enter all the network credentials by a setup wizard on first boot. If that happens, follow these steps to finish setting up your Raspberry Pi.

1. **Click Next** on the dialog box.



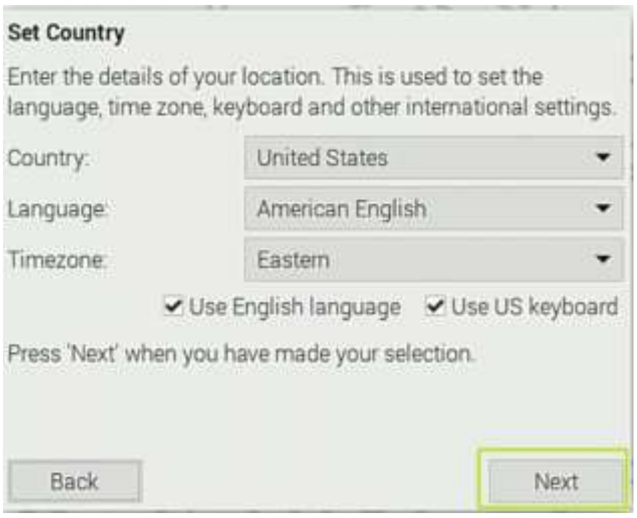
Welcome to the Raspberry Pi Desktop!

Before you start using it, there are a few things to set up.

Press 'Next' to get started.

If you are using a Bluetooth keyboard or mouse, put them into pairing mode and wait for them to connect.

2. **Set your country and and language** and click Next. The default choices may already be the correct ones.



Set Country

Enter the details of your location. This is used to set the language, time zone, keyboard and other international settings.

Country: United States

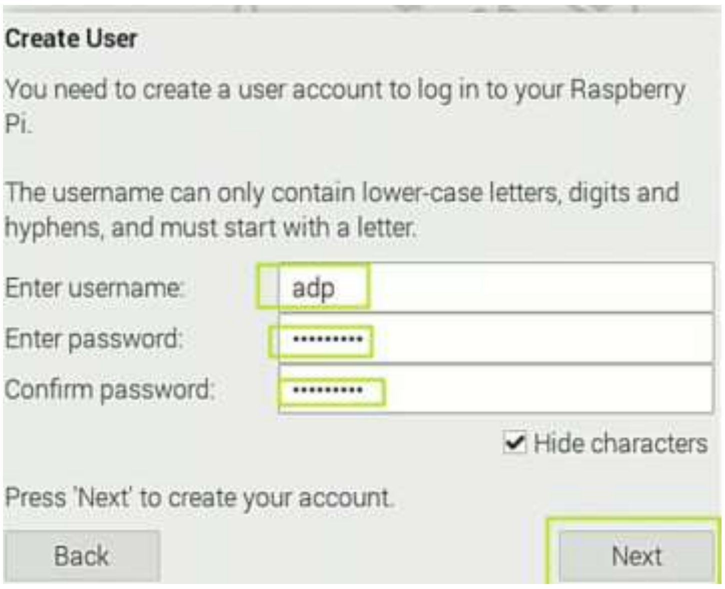
Language: American English

Timezone: Eastern

✓ Use English language ✓ Use US keyboard

Press 'Next' when you have made your selection.

3. **Enter a username and password** you wish to use for your primary login. **Click Next**.



Create User

You need to create a user account to log in to your Raspberry Pi.

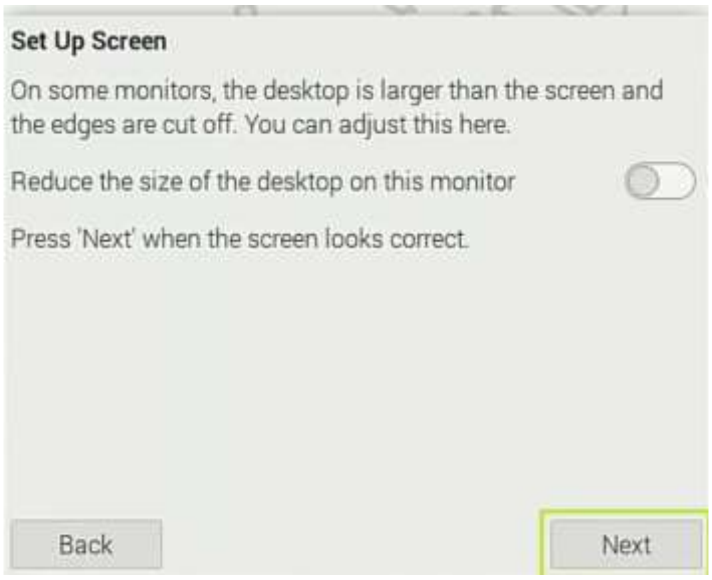
The username can only contain lower-case letters, digits and hyphens, and must start with a letter.

Enter username: adp
Enter password: ••••••••••
Confirm password: ••••••••••

✓ Hide characters

Press 'Next' to create your account.

4. **Toggle Reduce the size of the desktop" to on** if the borders of the desktop are cut off. Otherwise, just **click Next**.



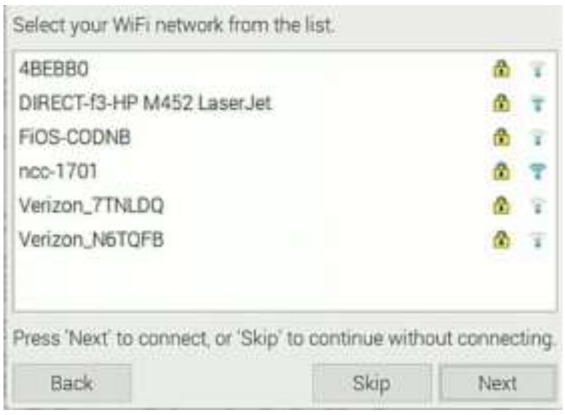
Set Up Screen

On some monitors, the desktop is larger than the screen and the edges are cut off. You can adjust this here.

Reduce the size of the desktop on this monitor

Press 'Next' when the screen looks correct.

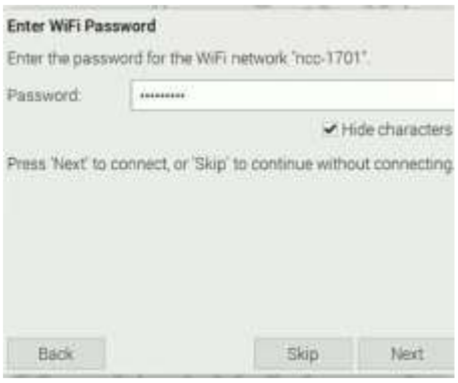
5. **Select the appropriate Wi-Fi network** on the screen after, provided that you are connecting via Wi-Fi. If you don't have Wi-Fi or are using Ethernet, you can skip this.


Select your WiFi network from the list.

4BEBB0
DIRECT-f3-HP M452 LaserJet
FiOS-CODNB
ncc-1701
Verizon_7TNLDQ
Verizon_N6TQFB

Press 'Next' to connect, or 'Skip' to continue without connecting.

6. **Enter your Wi-Fi password** (unless you were using Ethernet and skipped).


Enter WiFi Password

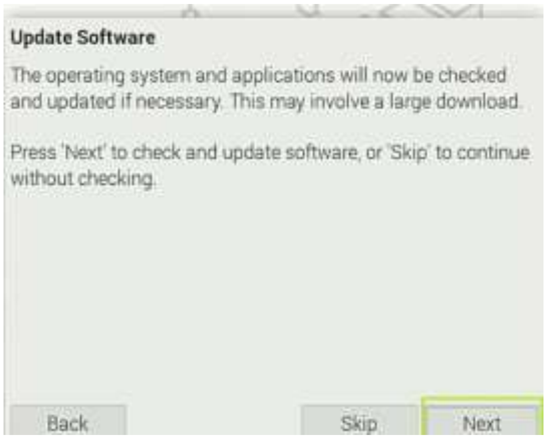
Enter the password for the WiFi network "ncc-1701".

Password: ••••••••••

✓ Hide characters

Press 'Next' to connect, or 'Skip' to continue without connecting.

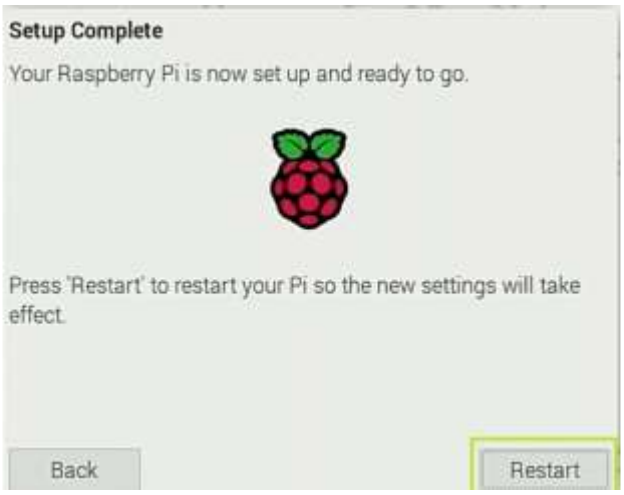
7. **Click Next** when prompted to Update Software. This will only work when you are connected to the Internet, and it can take several minutes. If you are not connected to the Internet, click Skip.


Update Software

The operating system and applications will now be checked and updated if necessary. This may involve a large download.

Press 'Next' to check and update software, or 'Skip' to continue without checking.

8. **Click Restart.**



Setup Complete

Your Raspberry Pi is now set up and ready to go.

Press 'Restart' to restart your Pi so the new settings will take effect.

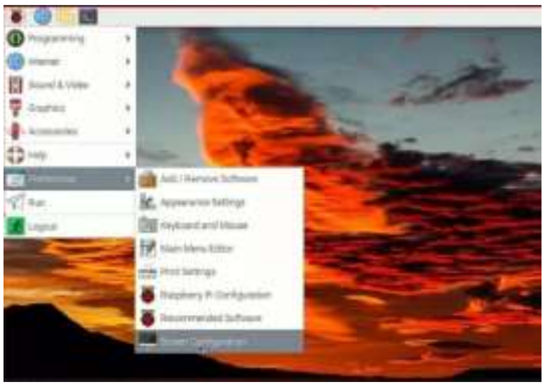
If you wish to change these settings later, you can find the region and password settings, along with many other options, by clicking on the Pi icon in the upper left corner of the screen and navigating to *Preferences -> Raspberry Pi Configuration*. You can configure Wi-Fi by clicking on the Wi-Fi / network icon on the taskbar.

(Image credit: Tom's Hardware)

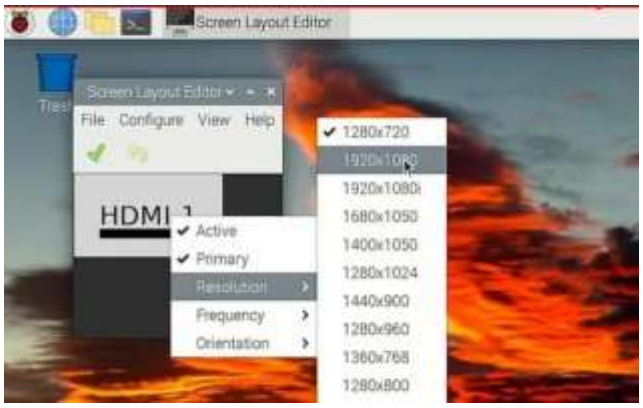
Changing Your Screen Resolution on Raspberry Pi

If you don't have enough desktop real estate, you may want to change your screen resolution to ensure that it matches what your display is capable of. If you are using a headless Pi and accessing it via VNC, you still probably want at least a 720p screen.
To change the Raspberry Pi resolution:

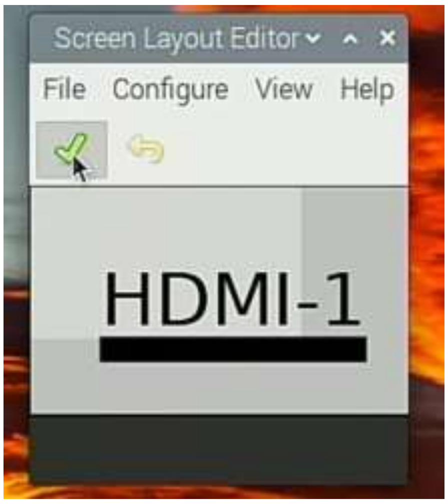
1. **Open the Screen configuration menu** by clicking on the Pi icon then selecting *Preferences -> Screen Configuration.*



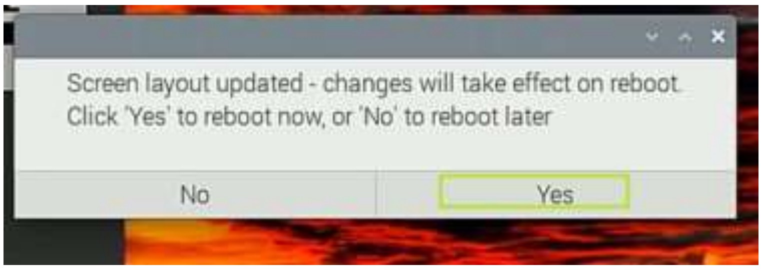
3. **Right Click on the HDMI box** and **select your Resolution** from the Resolution menu.



4. **Click the Check box.** The screen resolution will update.



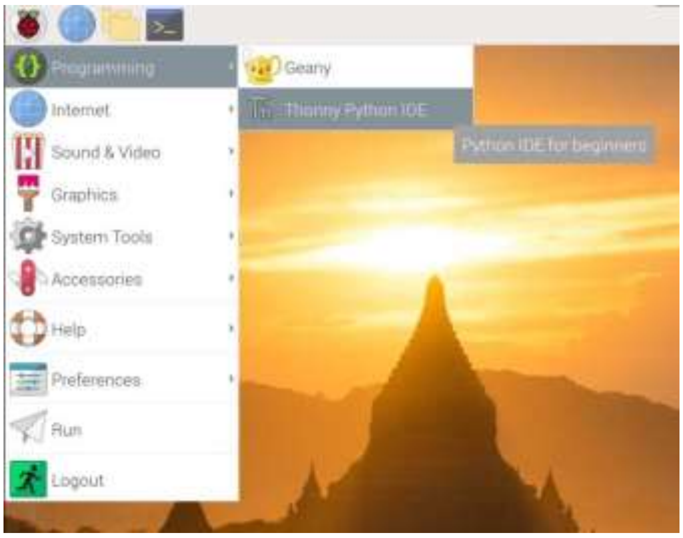
4. **Click Yes** to reboot.



Screen layout updated - changes will take effect on reboot.
Click 'Yes' to reboot now, or 'No' to reboot later

How to start Thonny IDE on Raspberry Pi

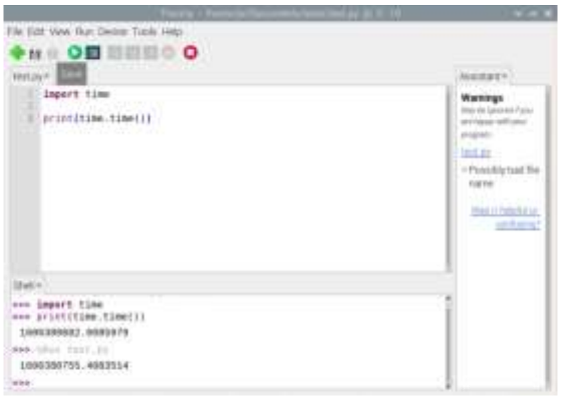
First, obviously, you need to install Raspberry Pi OS on a microSD card, for your Raspberry Pi board.



Click on the top left icon representing a raspberry > Programming > Thonny Python IDE.

Write complete Python programs on Thonny IDE

Running Python commands directly on the shell is of course not a viable solution for the mid/long term. You want to be able to write longer programs, save them, execute them, reopen them, etc.

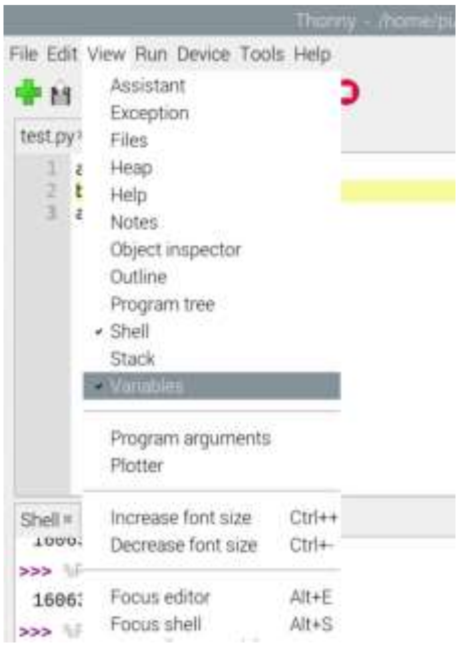


```
import time

print(time.time())
```

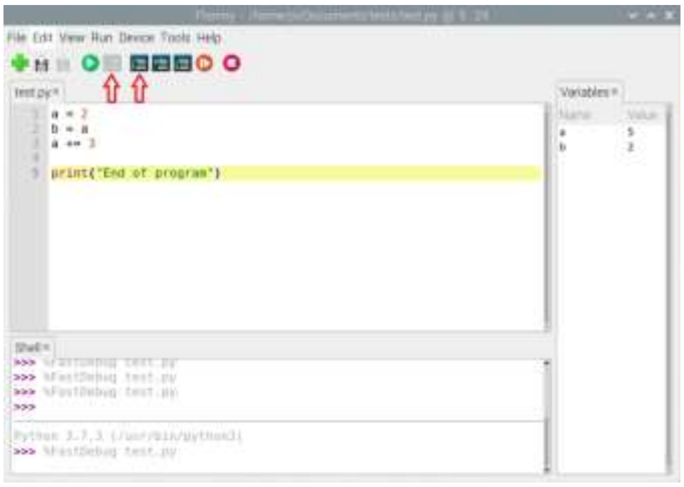
```
>>> import time
>>> print(time.time())
 [illegible]
>>> %Run test.py
 1600380755.4083514
>>>
```

Thonny IDE debugger

First, click on “View”. If you want to be able to see what are the values inside the variables of your program, make sure to click on “Variables”.



Then, click on the icon next to the “play” button to start the execution of your program.



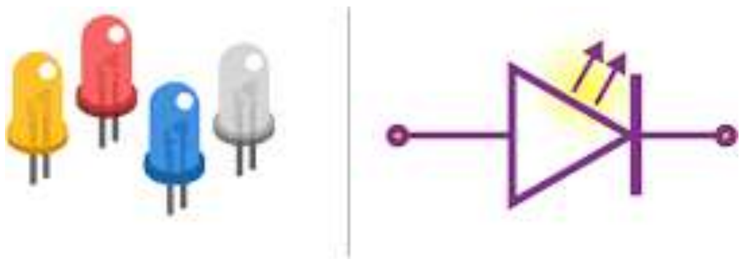
```
a = 2
b = a
a += 3

print("End of program")
```

Name	Value
a	5
b	2

```
>>> %FastDebug test.py
>>> %FastDebug test.py
>>> %FastDebug test.py
>>>

Python 3.7.3 (/usr/bin/python3)
>>> %FastDebug test.py
```

LED

It is most widely used semiconductor which emit either visible light or invisible infrared light when forward biased. Remote controls generate invisible light. A Light emitting diodes (LED) is an optical electrical energy into light energy when voltage is applied.



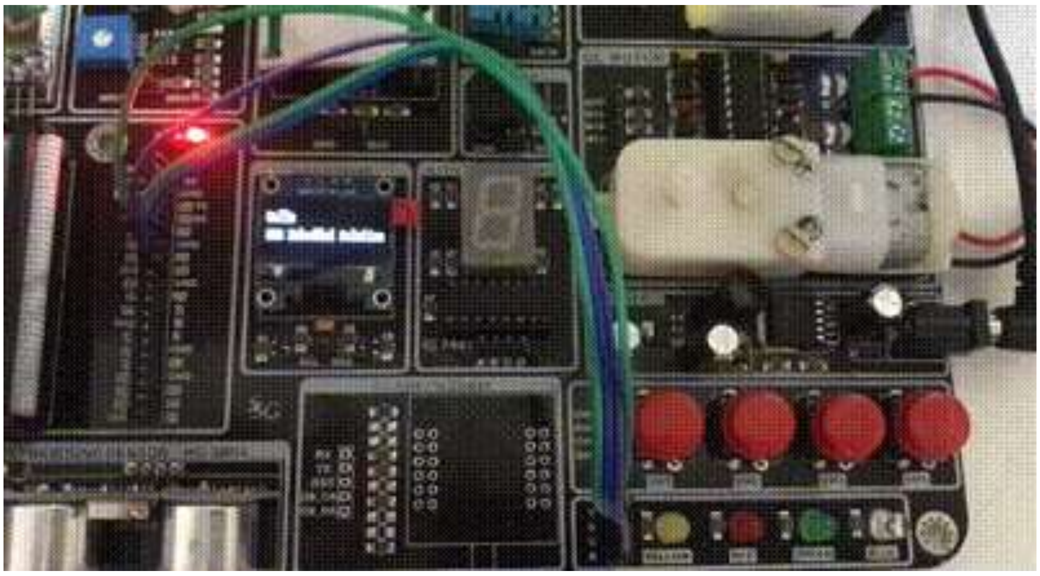
These are the applications of LEDs:

- Digital computers and calculators.
- Traffic signals and Burglar alarms systems.
- Camera flashes and automotive heat lamps
- Picture phones and digital watches.

LED Blinking Code

```
#!/usr/bin/python

import time

import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

GPIO.setup(17,GPIO.OUT)

GPIO.setup(27,GPIO.OUT)

GPIO.setup(23,GPIO.OUT)

GPIO.setup(24,GPIO.OUT)

#Turn LEDs on
```

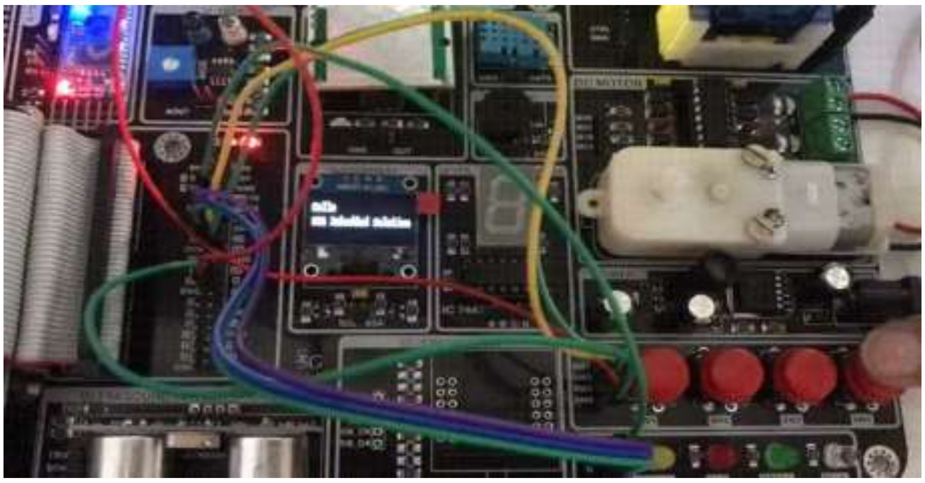
```
while True:
    GPIO.output(17,GPIO.HIGH)
    GPIO.output(27,GPIO.HIGH)
    GPIO.output(23,GPIO.HIGH)
    GPIO.output(24,GPIO.HIGH)
    time.sleep(1)
    print('led on')

#Turn LEDs off
    GPIO.output(17,GPIO.LOW)
    GPIO.output(27,GPIO.LOW)
    GPIO.output(23,GPIO.LOW)
    GPIO.output(24,GPIO.LOW)
    time.sleep(1)
    print('led off')
```



LED with Switch Code

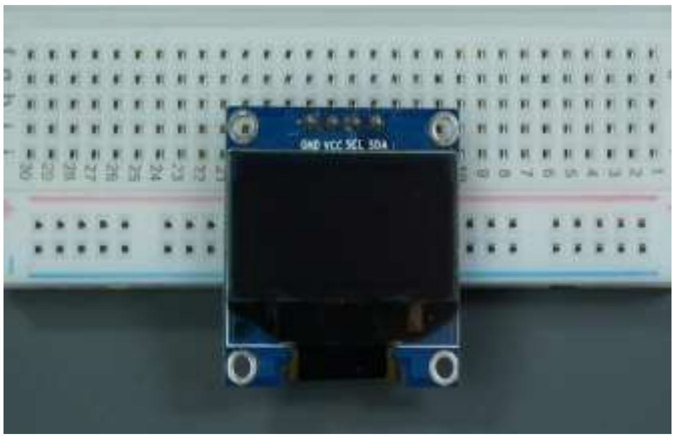
```
#CONNECT GPIO 18 TO LED PIN & GPIO 23 TO SWITCH
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

led1=4
led2=17
led3=27
led4=22

GPIO.setup(led1,GPIO.OUT)
GPIO.setup(led2,GPIO.OUT)
GPIO.setup(led3,GPIO.OUT)
GPIO.setup(led4,GPIO.OUT)
GPIO.setup(23,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(24,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(7,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(8,GPIO.IN,pull_up_down=GPIO.PUD_UP)
while True:
   GPIO.output(led1,GPIO.LOW)
   GPIO.output(led2,GPIO.LOW)
   GPIO.output(led3,GPIO.LOW)
   GPIO.output(led4,GPIO.LOW)
   input_state1=GPIO.input(23)
   input_state2=GPIO.input(24)
```

```
    input_state3=GPIO.input(7)
    input_state4=GPIO.input(8)
    if input_state1==False:
        GPIO.output(led1,GPIO.HIGH)
        print('BUTTON PRESSED')

    if input_state2==False:
        GPIO.output(led2,GPIO.HIGH)
        print('BUTTON PRESSED')

    if input_state3==False:
        GPIO.output(led3,GPIO.HIGH)
        print('BUTTON PRESSED')

    if input_state4==False:
        GPIO.output(led4,GPIO.HIGH)
        print('BUTTON PRESSED')
time.sleep(1)
```



OLED

OLED is the acronym for **Organic Light Emitting Diode**. OLED is a modern display technology used in a wide range of electronic display devices, such as TVs, monitors, laptops, smartphones, bulletin boards, stadium screens, etc.



OLED displays consist of organic semiconductor compounds that emit a bright light on the passage of electric current through them, and hence it is termed as OLED. Since, OLED displays can emit light on their own, thus they are considered as self-emissive types of display. There is no need of backlight panel with LEDs to illuminate the screen.

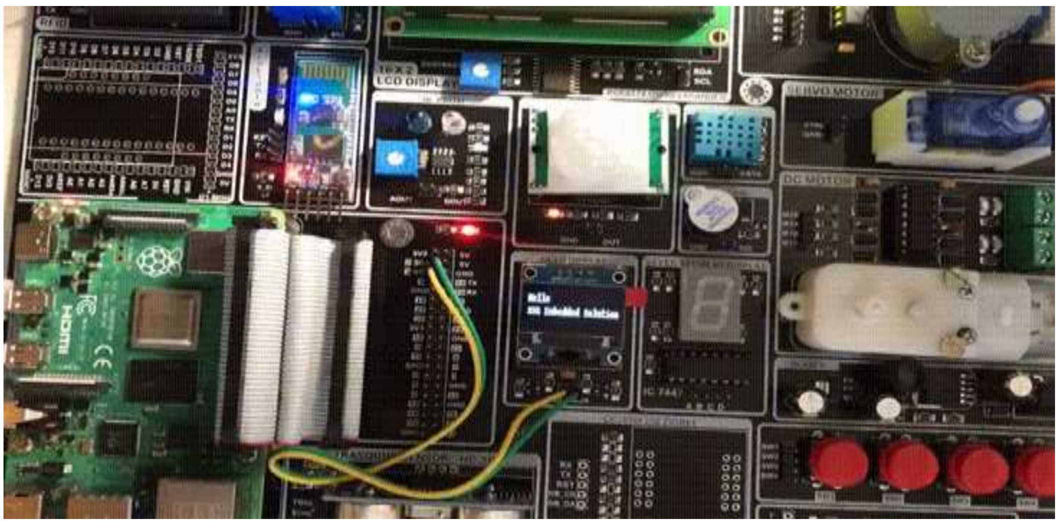
The primary advantages of OLED displays include better picture quality, relatively wider viewing angles, greater flexibility in design, compact size, faster response time, and low power consumption.

Code

```
import time

import Adafruit_GPIO.SPI as SPI

import Adafruit_SSD1306

from PIL import Image

from PIL import ImageDraw

from PIL import ImageFont
```

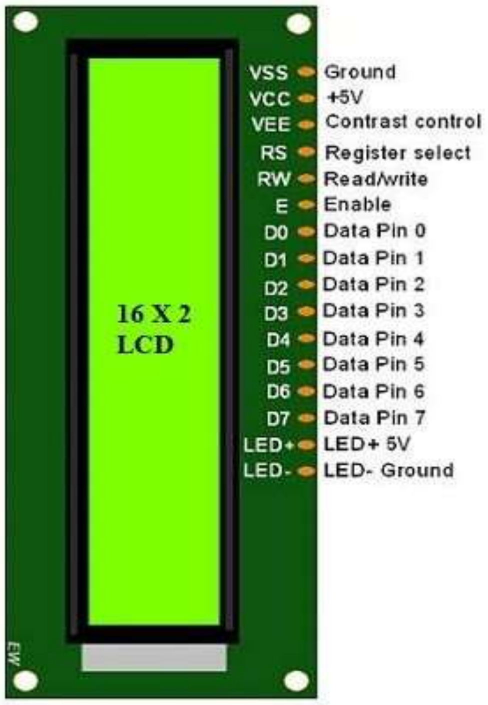
```
# Raspberry Pi pin configuration:

RST = 24

# Note the following are only used with SPI:

DC = 23

SPI_PORT = 0

SPI_DEVICE = 0

# Beaglebone Black pin configuration:

# RST = 'P9_12'

# Note the following are only used with SPI:

# DC = 'P9_15'

# SPI_PORT = 1

# SPI_DEVICE = 0

# 128x32 display with hardware I2C:

disp = Adafruit_SSD1306.SSD1306_128_32(rst=RST)

# Initialize library.

disp.begin()


# Clear display.

disp.clear()

disp.display()


# Create blank image for drawing.

# Make sure to create image with mode '1' for 1-bit color.

width = disp.width

height = disp.height

image = Image.new('1', (width, height))

draw = ImageDraw.Draw(image)
```

```
padding = 2
shape_width = 20
top = padding
bottom = height-padding
# Move left to right keeping track of the current x position for drawing shapes.
x = padding

# Load default font.
font = ImageFont.load_default()
draw.text((x, top),    'Hello',  font=font, fill=255,)
draw.text((x, top+10), 'SSG Embedded Solutions!', font=font, fill=255)

# Display image.
disp.image(image)
disp.display()
```



LCD

The term LCD stands for liquid crystal display. It is one kind of electronic display module used in an extensive range of applications like various circuits & devices like mobile phones, calculators, computers, TV sets, etc. These displays are mainly preferred for multi-segment light-emitting diodes and seven segments. The main benefits of using this module are inexpensive; simply programmable, animations, and there are no limitations for displaying custom characters, special and even animations, etc.

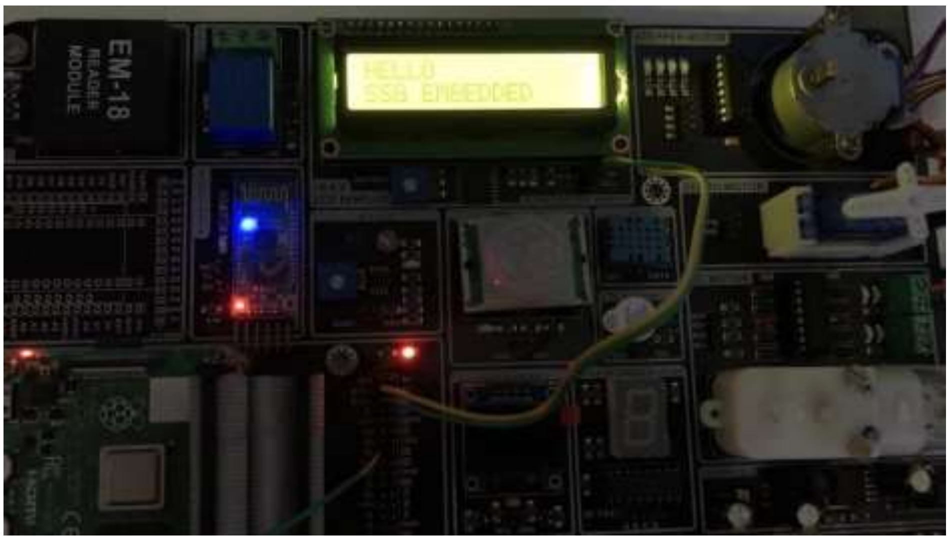


The features of this LCD mainly include the following.

- The operating voltage of this LCD is 4.7V-5.3V
- It includes two rows where each row can produce 16-characters.
- The utilization of current is 1mA with no backlight
- Every character can be built with a 5×8 pixel box
- The alphanumeric LCDs alphabets & numbers
- Is display can work on two modes like 4-bit & 8-bit

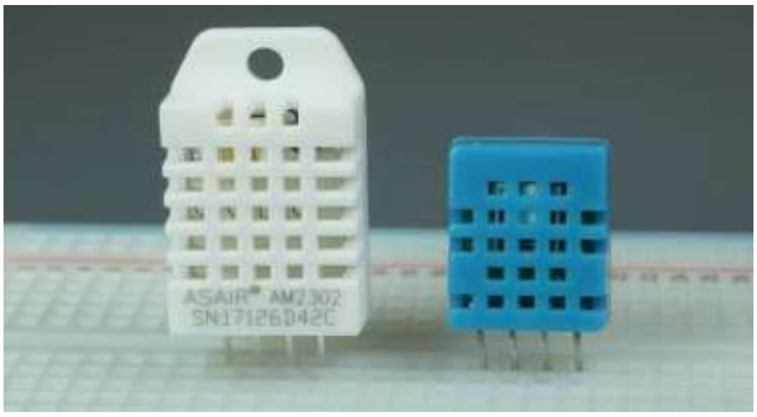
Code

```
from signal import signal, SIGTERM,SIGHUP, pause

from rpi_lcd import LCD

lcd = LCD()

def safe_exit(signum, frame):

    exit(1)

try:

    signal(SIGTERM, safe_exit)

    signal(SIGHUP, safe_exit)

    lcd.text("HELLO",1)

    lcd.text("SSG EMBEDDED SOLUTION",2)

    pause()

except KeyboardInterrupt:

    pass

finally:

    lcd.clear()
```



DHT11

The DHT11 and DHT22 sensors are used to measure temperature and relative humidity. These sensors contain a chip that does analog to digital conversion and spit out a digital signal with the temperature and humidity. This makes them very easy to use with any microcontroller.



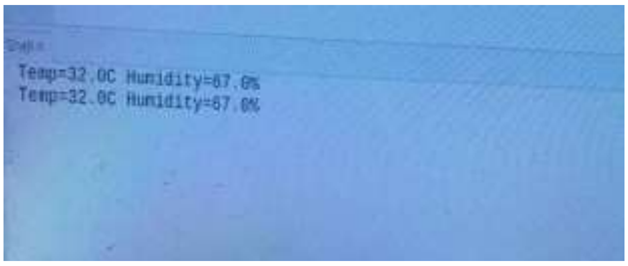
The DHT22 sensor has a better resolution and a wider temperature and humidity measurement range. However, it is a bit more expensive, and you can only request readings with 2 seconds interval. The DHT33 has a smaller range and it’s less accurate. However, you can request sensor readings every second. It’s also a bit cheaper.

Code

```
import time

import Adafruit_GPIO.SPI as SPI

import Adafruit_SSD1306


from PIL import Image

from PIL import ImageDraw

from PIL import ImageFont
```

```
# Raspberry Pi pin configuration:

RST = 24

# Note the following are only used with SPI:

DC = 23

SPI_PORT = 0

SPI_DEVICE = 0


# Beaglebone Black pin configuration:

# RST = 'P9_12'

# Note the following are only used with SPI:

# DC = 'P9_15'

# SPI_PORT = 1

# SPI_DEVICE = 0


# 128x32 display with hardware I2C:

disp = Adafruit_SSD1306.SSD1306_128_32(rst=RST)




# Initialize library.

disp.begin()


# Clear display.

disp.clear()

disp.display()
```

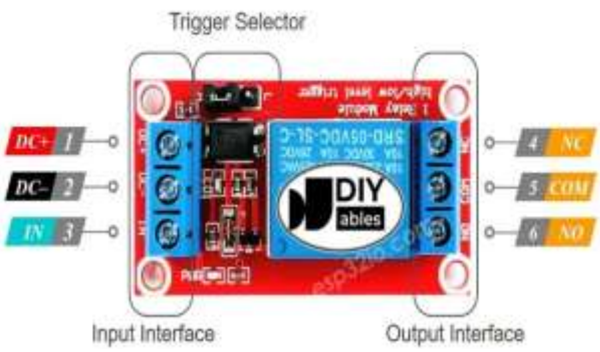
```
# Create blank image for drawing.

# Make sure to create image with mode '1' for 1-bit color.

width = disp.width

height = disp.height

image = Image.new('1', (width, height))

draw = ImageDraw.Draw(image)


padding = 2

shape_width = 20

top = padding

bottom = height-padding

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

x = padding


# Load default font.

font = ImageFont.load_default()

draw.text((x, top),    'Hello',  font=font, fill=255,)

draw.text((x, top+10), 'SSG Embedded Solutions!', font=font, fill=255)


# Display image.

disp.image(image)

disp.display()
```



```
Temp=32.0C Humidity=67.0%
Temp=32.0C Humidity=67.0%
```

Relay

Relays are the switches that aim at closing and opening the circuits electronically as well as electromechanically. It controls the opening and closing of the circuit contacts of an electronic circuit. When the relay contact is open (NO), the relay isn't energized with the open contact. However, if it is closed (NC), the relay isn't energized given the closed contact. However, when energy (electricity or charge) is supplied, the states are prone to change.

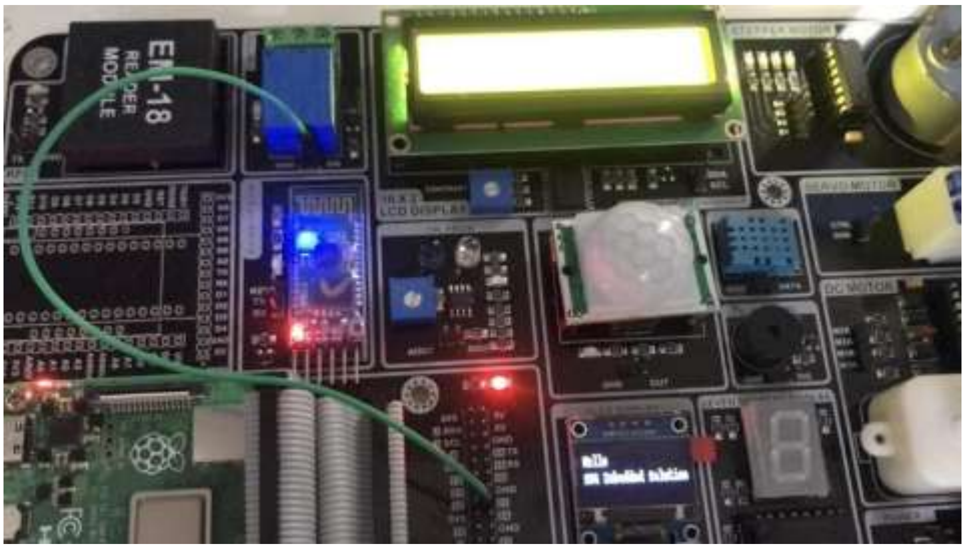


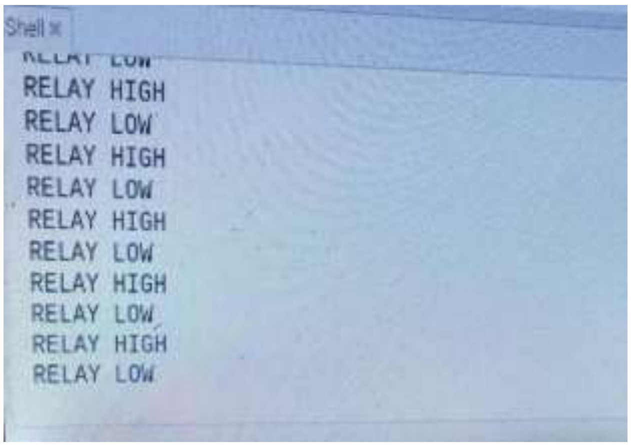
Relays are normally used in the control panels, manufacturing, and building automation to control the power along with switching the smaller current values in a control circuit. However, the supply of amplifying effect can help control the large amperes and voltages because if low voltage is applied to the relay coil, a large voltage can be switched by the contacts.

Code

```
# CONNECT BOARD 16 OR GPIO 23 WITH RELAY SIG
import RPi.GPIO as GPIO
import time


GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
buzzer=23
GPIO.setup(buzzer,GPIO.OUT)
```

```
while True:
    GPIO.output(buzzer,GPIO.HIGH)
    print("RELAY HIGH")
    time.sleep(0.5)
    GPIO.output(buzzer,GPIO.LOW)
    print("RELAY LOW")
    time.sleep(0.5)
```





```
Shell
RELAY LOW
RELAY HIGH
RELAY LOW
RELAY HIGH
RELAY LOW
RELAY HIGH
RELAY LOW
RELAY HIGH
RELAY LOW
RELAY HIGH
RELAY LOW
```

Buzzer

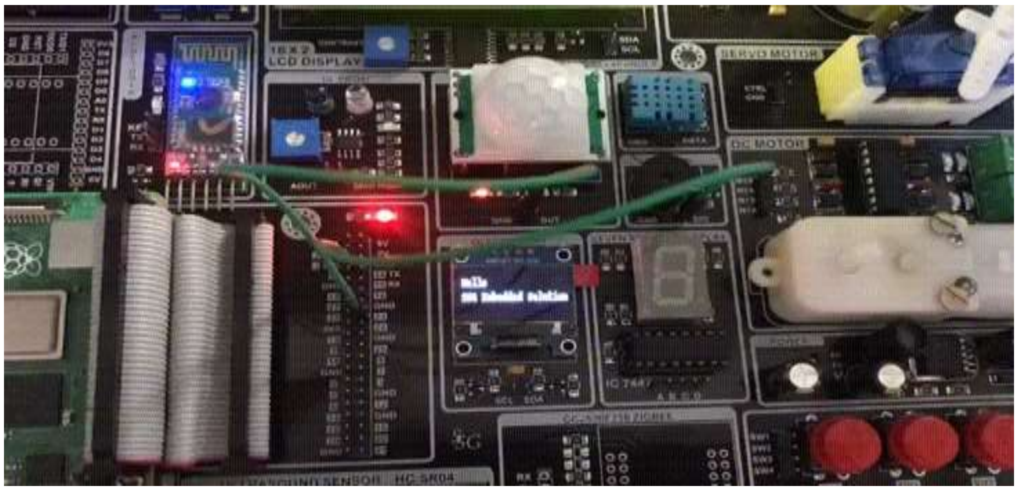
A **buzzer** is an electronic device that generates sound by converting electrical energy into sound energy. It typically consists of a piezoelectric crystal, which expands and contracts when an alternating current is applied to it, creating sound waves.

Buzzers are commonly used in a wide range of applications such as alarms, timers, and warning systems. They can also be used in electronic devices such as mobile phones, computers, and other electronic devices to generate different sounds and tones.

Code

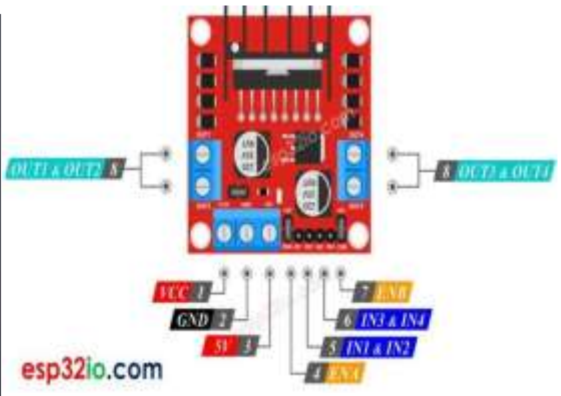
```
# connect BOARD 16 WITH BUZZER
import RPi.GPIO as GPIO
import time


GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
buzzer=23
```

```
GPIO.setup(buzzer,GPIO.OUT)
while True:
   GPIO.output(buzzer,GPIO.HIGH)
   time.sleep(2)
   GPIO.output(buzzer,GPIO.LOW)
   time.sleep(2)
```



DC Motor

DC motor uses Direct Current (electrical energy) to produce mechanical movement i.e. rotational movement. When it converts electrical energy into mechanical energy then it is called as DC motor and when it converts mechanical energy into electrical energy then it is called as DC generator.



The working principle of DC motor is based on the fact that when a current carrying conductor is placed in a magnetic field, it experiences a mechanical force and starts rotating. Its direction of rotation depends upon Fleming’s Left Hand Rule.

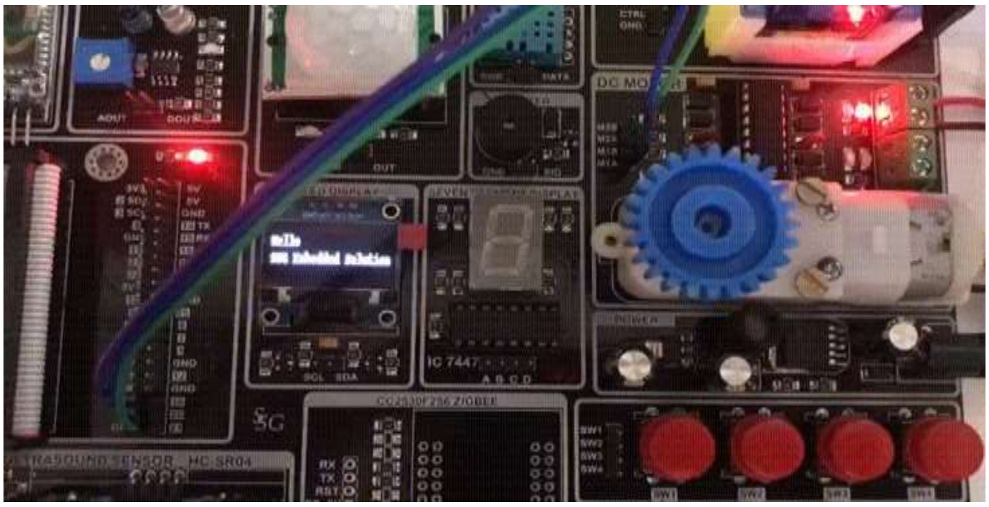
DC motors are used in many applications like robot for movement control, toys, quadcopters, CD/DVD disk drive in PCs/Laptops etc.

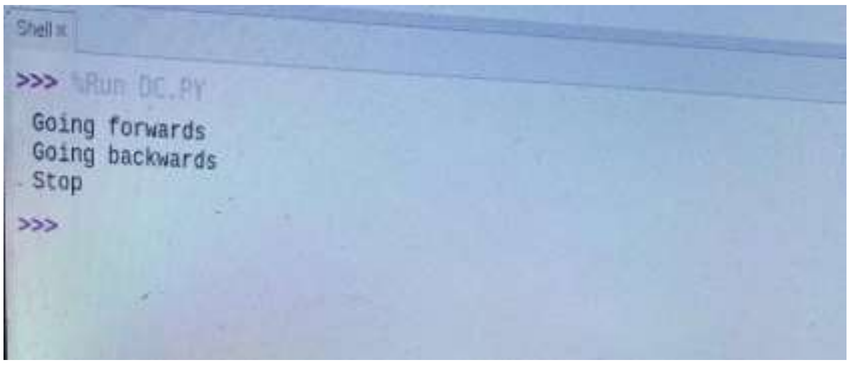
Code

```
# CONNECT GPIO 21 TO M2A & GPIO 20 TO M2B

import RPi.GPIO as GPIO

from time import sleep


# Pins for Motor Driver Inputs

Motor1A = 21
```

```
Motor1B = 20

Motor1E = 16


def setup():

    GPIO.setwarnings(False)

    GPIO.setmode(GPIO.BCM)              # GPIO Numbering

    GPIO.setup(Motor1A,GPIO.OUT)  # All pins as Outputs

    GPIO.setup(Motor1B,GPIO.OUT)

    GPIO.setup(Motor1E,GPIO.OUT)


def loop():

    # Going forwards

    GPIO.output(Motor1A,GPIO.HIGH)

    GPIO.output(Motor1B,GPIO.LOW)

    GPIO.output(Motor1E,GPIO.HIGH)

    print("Going forwards")


    sleep(5)

    # Going backwards

    GPIO.output(Motor1A,GPIO.LOW)

    GPIO.output(Motor1B,GPIO.HIGH)

    GPIO.output(Motor1E,GPIO.HIGH)

    print("Going backwards")


    sleep(5)

    # Stop

    GPIO.output(Motor1E,GPIO.LOW)
```

```
    GPIO.output(Motor1B,GPIO.LOW)

    print("Stop")


def destroy():

    GPIO.cleanup()


if __name___== '__main__':        # Program start from here

    setup()

    try:

            loop()

    except KeyboardInterrupt:

        destroy()
```

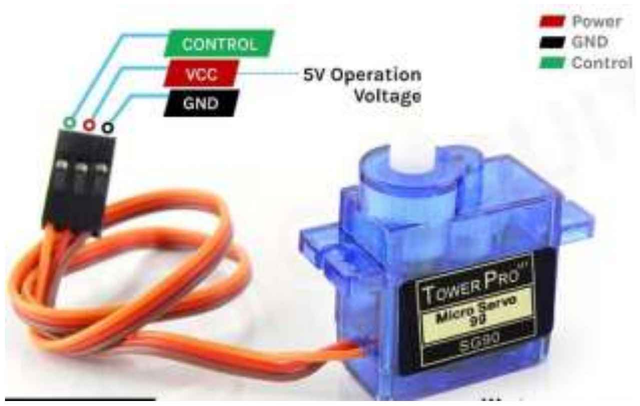




```
Shell
>>> %Run DC.PY
 Going forwards
 Going backwards
 Stop
>>>
```

Servo Motor

Servo motor is an electrical device which can be used to rotate objects (like robotic arm) precisely.Servo motor consists of DC motor with error sensing negative feedback mechanism. This allows precise control over angular velocity and position of motor. In some cases, AC motors are used.



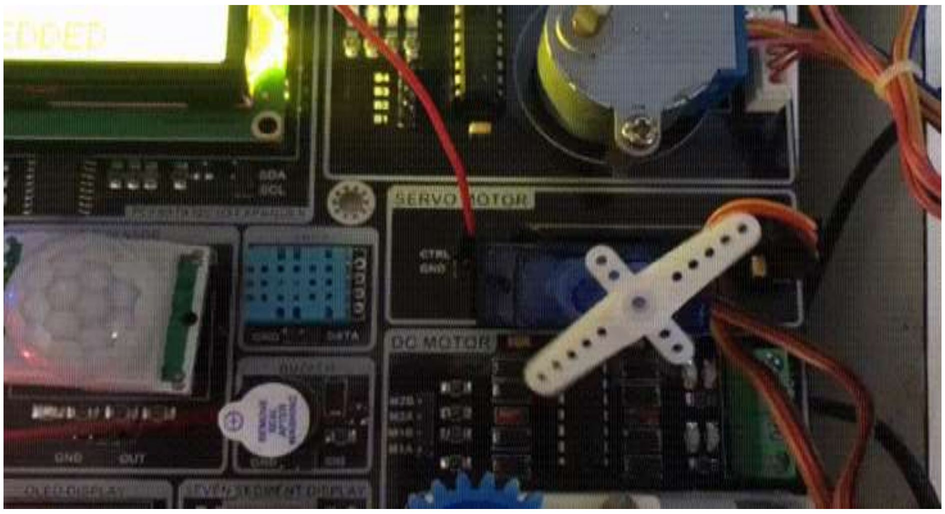
It is a closed loop system where it uses negative feedback to control motion and final position of the shaft.It is not used for continuous rotation like conventional AC/DC motors.It has rotation angle that varies from 0° to 380°.

Code

```
# CONNECT BOARD 3 OR GPIO 2 TO SERVO CTRL

import RPi.GPIO as GPIO

import time


control = [5,5.5,6,6.5,7,7.5,8,8.5,9,9.5,10]


servo = 3
```

```
GPIO.setmode(GPIO.BOARD)


GPIO.setup(servo,GPIO.OUT)
# in servo motor,
# 1ms pulse for 0 degree (LEFT)
# 1.5ms pulse for 90 degree (MIDDLE)
# 2ms pulse for 180 degree (RIGHT)


# so for 50hz, one frequency is 20ms
# duty cycle for 0 degree = (1/20)*100 = 5%
# duty cycle for 90 degree = (1.5/20)*100 = 7.5%
# duty cycle for 180 degree = (2/20)*100 = 10%


p=GPIO.PWM(servo,50)# 50hz frequency


p.start(2.5)# starting duty cycle ( it set the servo to 0 degree )



try:
       while True:
           for x in range(11):
             p.ChangeDutyCycle(control[x])
             time.sleep(0.05)
             print ('x')


           for x in range(9,0,-1):
             p.ChangeDutyCycle(control[x])
```

```
            time.sleep(0.05)

            print ('x')


except KeyboardInterrupt:

    GPIO.cleanup()
```



Stepper Motor

Stepper Motor is a brushless DC Motor. Control signals are applied to stepper motor to rotate it in steps.

Its speed of rotation depends upon rate at which control signals are applied. There are various stepper motors available with minimum required step angle.

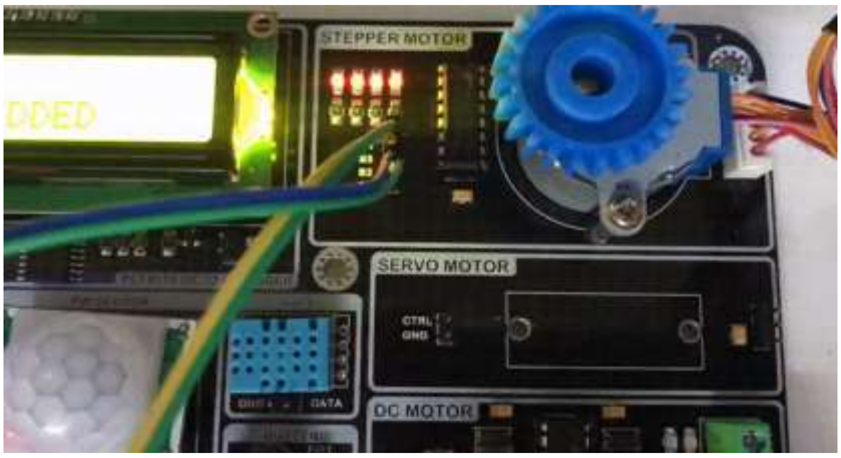
Stepper motor is made up of mainly two parts, a stator and rotor. Stator is of coil winding and rotor is mostly permanent magnet or ferromagnetic material.

Step angle is the minimum angle that stepper motor will cover within one move/step. Number of steps required to complete one rotation depends upon step angle. Depending upon stepper motor configuration, step angle varies e.g. 0.72°, 3.8°, 3.75°, 7.5°, 35° etc.

Code

```
# connect BOARD 7,11,13 &15 OR GPIO 4,17,27 & 22 TO B1,B2,B3 &B4 respectively

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BCM)

#GPIO.setmode(GPIO.BOARD)

GPIO.setwarnings(False)
```

```
ControlPin=[4,17,27,22]

for pin in ControlPin:
    GPIO.setup(pin,GPIO.OUT)
    GPIO.output(pin,1)

seq=  [[1,1,0,0],
       [0,1,1,0],
       [0,0,1,1],
       [1,0,0,1],
 ]

rotationNeeded=0
rotationCount=0
while(1):
    rotationNeeded==0
    print('/n')
    userInput=input('e-exit,enter rotation number')
    print('/n')
    if userInput=='e':
        break;

    rotationNeeded=int (userInput)
    rotationCount=50* rotationNeeded
    for i in range(rotationCount):
        for fullstep in range(4):
            for pin in range(4):
                GPIO.output(ControlPin[pin],seq[fullstep][pin])
                time.sleep(0.001)
GPIO.cleanup()
```



IR Sensor

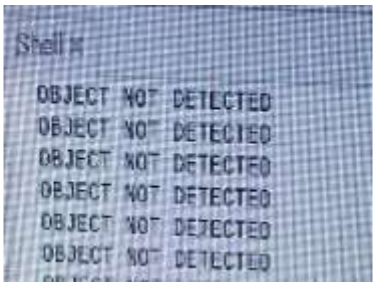
IR sensor is an electronic device, that emits the light in order to sense some object of the surroundings. An **IR sensor** can measure the heat of an object as well as detects the motion. Usually, in the **infrared spectrum**, all the objects radiate some form of thermal radiation. These types of radiations are invisible to our eyes, but infrared sensor can detect these radiations.

The emitter is simply an IR LED **(Light Emitting Diode**) and the detector is simply an IR photodiode . Photodiode is sensitive to IR light of the same wavelength which is emitted by the IR LED. When IR light falls on the photodiode, the resistances and the output voltages will change in proportion to the magnitude of the IR light received.

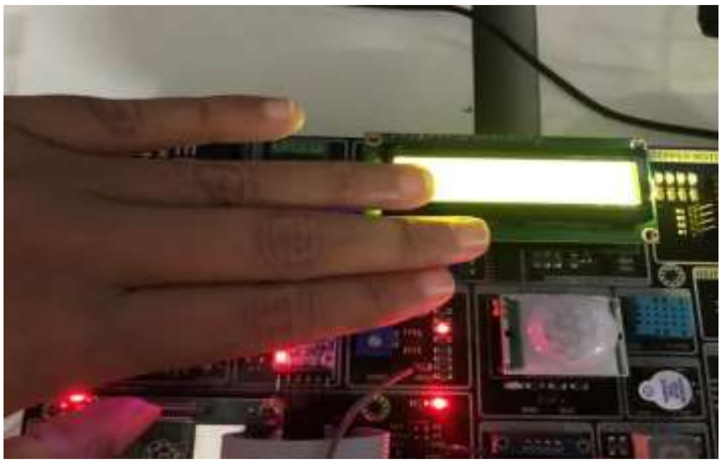
Code

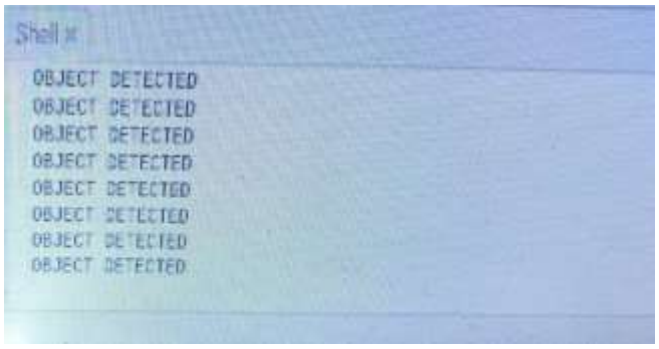
```
# connect BOARD PIN 7  OR GPIO 4 TO DOUT PIN of IR PROXI

# CONNECT BOARD PIN 5 OR GPIO 3 TO LED PIN

import RPi.GPIO as GPIO

import time


GPIO.setmode(GPIO.BOARD)


GPIO.setup(7,GPIO.IN)

GPIO.setup(5,GPIO.OUT)
```

```
while True:
    val=GPIO.input(7)
#     print(val)
    if val==1:
        GPIO.output(5,GPIO.HIGH)
        print('OBJECT DETECTED')
    else:
        GPIO.output(5,GPIO.LOW)
        print('OBJECT NOT DETECTED')
sleep.time(100)
```



```
Shell ×
OBJECT NOT DETECTED
OBJECT NOT DETECTED
OBJECT NOT DETECTED
OBJECT NOT DETECTED
OBJECT NOT DETECTED
OBJECT NOT DETECTED
```

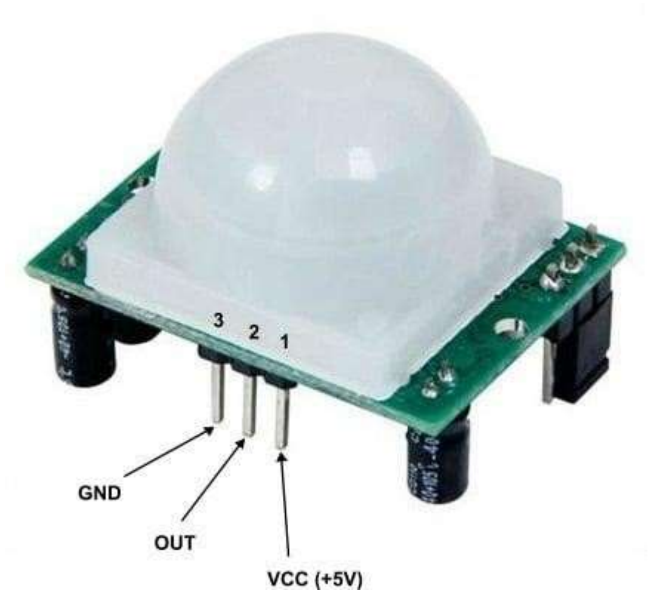




```
Shell ×
OBJECT DETECTED
OBJECT DETECTED
OBJECT DETECTED
OBJECT DETECTED
OBJECT DETECTED
OBJECT DETECTED
OBJECT DETECTED
OBJECT DETECTED
```

PIR Sensor

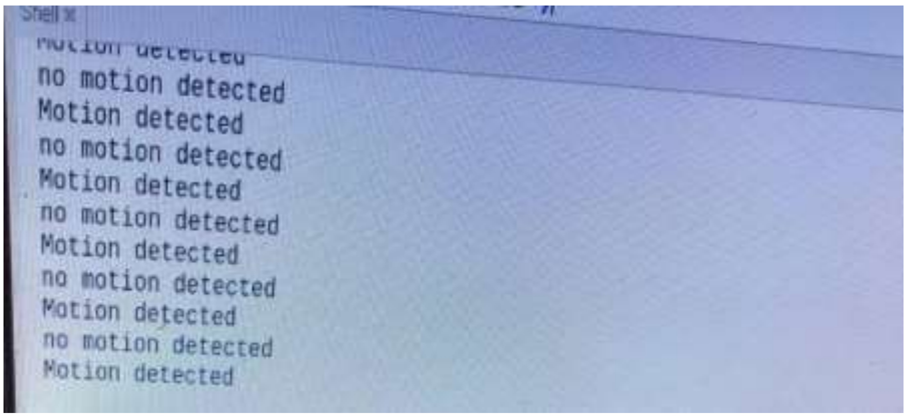
passive infrared sensor is an electronic sensor that measures infrared light radiating from objects. PIR sensors mostly used in PIR-based motion detectors. Also, it used in security alarms and automatic lighting applications. The below image shows a typical pin configuration of the PIR sensor, which is quite simple to understand the pinouts. The PIR sensor consists of 3 pins,



- Pin1 corresponds to the drain terminal of the device, which connected to the positive supply 5V DC.
- Pin2 corresponds to the source terminal of the device, which connects to the ground terminal via a 100K or 47K resistor. The Pin2 is the output pin of the sensor. The pin 2 of the sensor carries the detected IR signal to an amplifier from the
- Pin3 of the sensor connected to the ground.

Code

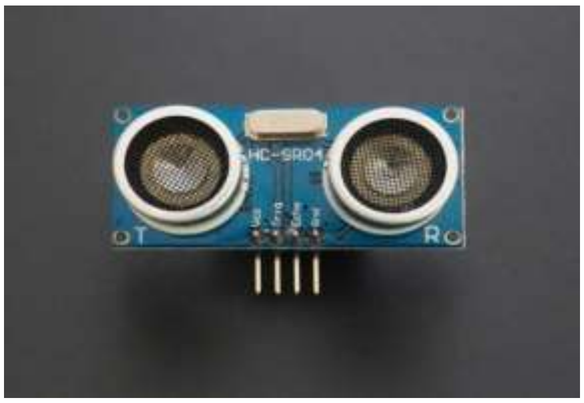
```
# connect GPIO 23 TO PIR OUT

# connect GPIO 3 TO Led pin

from gpiozero import MotionSensor

import RPi.GPIO as GPIO
import time
from time import sleep
pir = MotionSensor(23)
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(3,GPIO.OUT)
while True:
        pir.wait_for_motion()
        GPIO.output(3, 1)
        print("Motion detected")
#       time.sleep(5)
        pir.wait_for_no_motion()
        GPIO.output(3, 0)
        print("no motion detected")
```



```
Shell
Motion detected
no motion detected
Motion detected
no motion detected
Motion detected
no motion detected
Motion detected
no motion detected
Motion detected
no motion detected
Motion detected
```

Ultrasonic Sensor

Ultrasonic Module HC-SR04 works on the principle of SONAR and RADAR systems. It can be used to determine the distance of an object in the range of 2 cm – 400 cm. An ultrasonic sensor generates high-frequency sound (ultrasound) waves. When this ultrasound hits the object, it reflects as echo which is sensed by the receiver



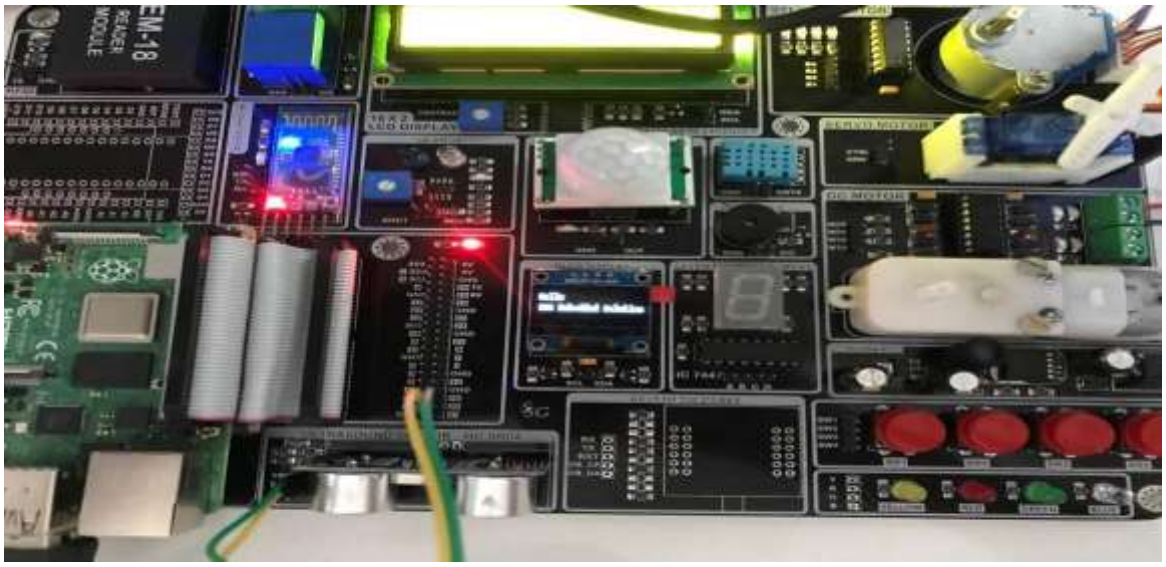
HC-SR-04 has an ultrasonic transmitter, receiver and control circuit.

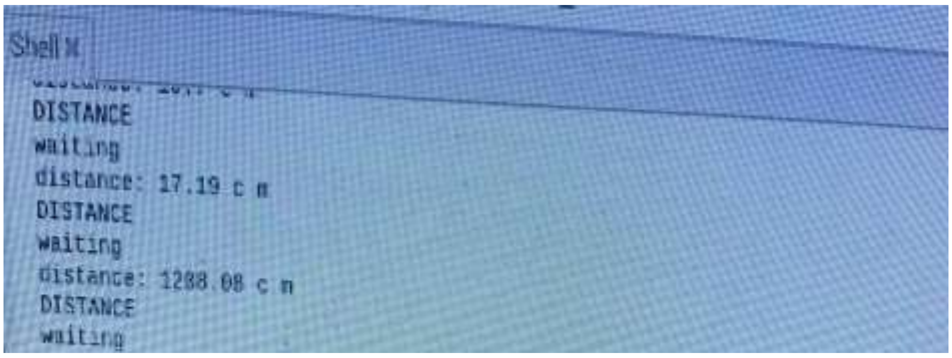
In the ultrasonic module HCSR04, we have to give trigger pulse, so that it will generate ultrasound of frequency 40 kHz. After generating ultrasound i.e. 8 pulses of 40 kHz, it makes echo pin high. Echo pin remains high until it does not get the echo sound back. So the width of echo pin will be the time for sound to travel to the object and return back. Once we get the time we can calculate distance, as we know the speed of sound.

Code

```
# connect BOARD 38 TO ECHO &40 TO TRIG

import RPi.GPIO as GPIO

import time
```

```
TRIG=21
ECHO=20

GPIO.setmode(GPIO.BCM)
while True:
    print('DISTANCE')
    GPIO.setup(TRIG,GPIO.OUT)
    GPIO.setup(ECHO,GPIO.IN)
    GPIO.output(TRIG,False)
    print('waiting')
    time.sleep(0.5)

    GPIO.output(TRIG,True)
    time.sleep(0.00001)
    GPIO.output(TRIG,False)

    while GPIO.input(ECHO)==0:
        pluse_start=time.time()

    while GPIO.input(ECHO)==1:
        pluse_end=time.time()

    pluse_duration=pluse_end-pluse_start
    distance=pluse_duration*17150
    distance=round(distance,2)
    print('distance:',distance,'c m')
    time.sleep(0.5)
```

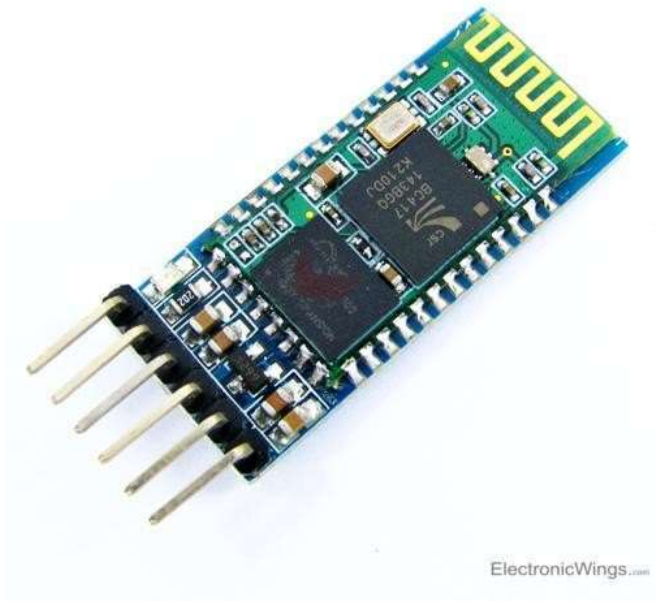




```
DISTANCE
waiting
distance: 17.19 c m
DISTANCE
waiting
distance: 1288.08 c m
DISTANCE
waiting
```

Bluetooth

It is used for many applications like wireless headset, game controllers, wireless mouse, wireless keyboard, and many more consumer applications.It has range up to <100m which depends upon transmitter and receiver, atmosphere, geographic & urban conditions.



It is IEEE 802.15.1 standardized protocol, through which one can build wireless Personal Area Network (PAN). It uses frequency-hopping spread spectrum (FHSS) radio technology to send data over air.

It uses serial communication to communicate with devices. It communicates with microcontroller using serial port (USART).

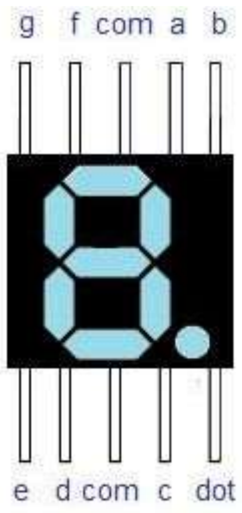
Code

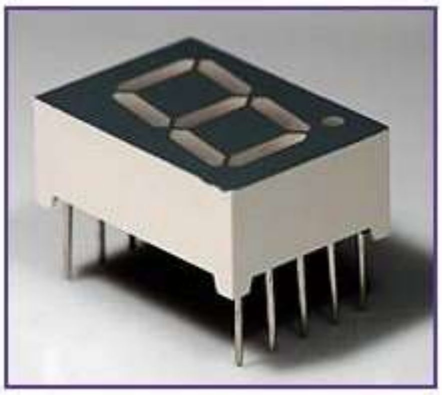
```
# connect RX & TX of bluetooth with TX & RX of RPI4 and LED PIN to GPIO  27,brfore that Set up Raspberry Pi for Serial Communication

# by entering the  command sudo raspi-config in terminal-->select interface options--->serial option--->NO-->YES-->sudo reboot
```

```
#sudo raspi-config

import RPi.GPIO as GPIO

import serial

value = serial.Serial("/dev/ttyS0",baudrate=9600)



GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

GPIO.setup(27,GPIO.OUT)



while True:

    data = str (int(value.readline(),2))

    print(data)

    if data =="1":

        GPIO.output(27,GPIO.HIGH)

        print("LED1:ON")

    else:

        GPIO.output(27,GPIO.LOW)

        print("LED1:OFF")
```

Seven Segment

Seven segment displays are important display units in Electronics and widely used to display numbers from 0 to 9. It can also display some character alphabets like A,B,C,H,F,E etc. It's the simplest unit to display numbers and characters. It just consists 8 LEDs, each LED used to illuminate one segment of unit and the 8th LED used to illuminate DOT in 7 segment display. We can refer each segment as a LINE, as we can see there are 7 lines in the unit, which are used to display a number/character. We can refer each line/segment "a,b,c,d,e,f,g" and for dot character we will use "h". There are 10 pins, in which 8 pins are used to refer a,b,c,d,e,f,g and h/dp, the two middle pins are common anode/cathode of all he LEDs. These common anode/cathode are internally shorted so we need to connect only one COM pin.





There are two types of 7 segment displays: Common Anode and Common Cathode:

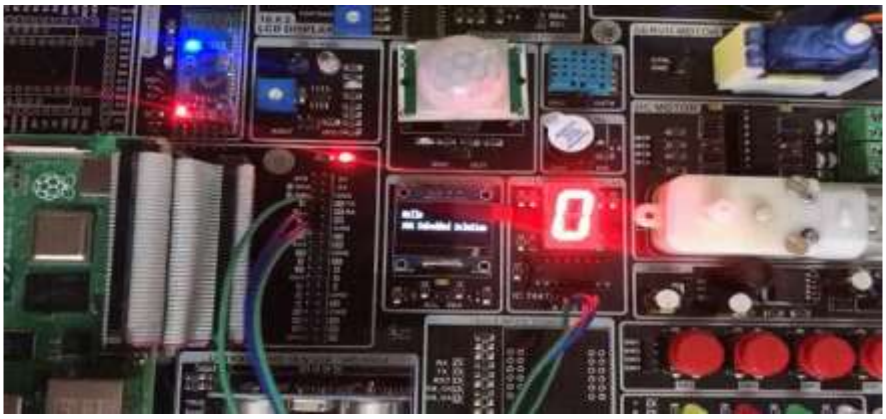
Common Anode: In this all the Negative terminals (cathode) of all the 8 LEDs are connected together (see diagram below), named as COM. And all the positive terminals are left alone.

Common Cathode: In this all the positive terminals (Anodes) of all the 8 LEDs are connected together, named as COM. And all the negative thermals are left alone.

Code

```
# CONNECT GPIO 4,17,27 & 22 TO A,B,C & D

import RPi.GPIO as GPIO

import time


BCD_PIN0 = 4

BCD_PIN1 = 17

BCD_PIN2 = 27

BCD_PIN3 = 22


GPIO.setwarnings(False)

# ["0000","0001","0010","0011","0100","0101","0110","0111","1000","1001"]


GPIO.setmode(GPIO.BCM)

GPIO.setup(BCD_PIN0,  GPIO.OUT)

GPIO.setup(BCD_PIN1,  GPIO.OUT)

GPIO.setup(BCD_PIN2,  GPIO.OUT)

GPIO.setup(BCD_PIN3,  GPIO.OUT)


while True:

# 0

    GPIO.output(BCD_PIN0,GPIO.LOW)

    GPIO.output(BCD_PIN1,GPIO.LOW)

    GPIO.output(BCD_PIN2,GPIO.LOW)

    GPIO.output(BCD_PIN3,GPIO.LOW)
```

```
    time.sleep(1)

#1
    GPIO.output(BCD_PIN0,GPIO.HIGH)
    GPIO.output(BCD_PIN1,GPIO.LOW)
    GPIO.output(BCD_PIN2,GPIO.LOW)
    GPIO.output(BCD_PIN3,GPIO.LOW)
    time.sleep(1)

#2
    GPIO.output(BCD_PIN0,GPIO.LOW)
    GPIO.output(BCD_PIN1,GPIO.HIGH)
    GPIO.output(BCD_PIN2,GPIO.LOW)
    GPIO.output(BCD_PIN3,GPIO.LOW)
    time.sleep(1)

#3
    GPIO.output(BCD_PIN0,GPIO.HIGH)
    GPIO.output(BCD_PIN1,GPIO.HIGH)
    GPIO.output(BCD_PIN2,GPIO.LOW)
    GPIO.output(BCD_PIN3,GPIO.LOW)
    time.sleep(1)

#4
    GPIO.output(BCD_PIN0,GPIO.LOW)
    GPIO.output(BCD_PIN1,GPIO.LOW)
    GPIO.output(BCD_PIN2,GPIO.HIGH)
    GPIO.output(BCD_PIN3,GPIO.LOW)
    time.sleep(1)

#5
    GPIO.output(BCD_PIN0,GPIO.HIGH)
    GPIO.output(BCD_PIN1,GPIO.LOW)
    GPIO.output(BCD_PIN2,GPIO.HIGH)
    GPIO.output(BCD_PIN3,GPIO.LOW)
    time.sleep(1)

# 6
    GPIO.output(BCD_PIN0,GPIO.LOW)
    GPIO.output(BCD_PIN1,GPIO.HIGH)
    GPIO.output(BCD_PIN2,GPIO.HIGH)
    GPIO.output(BCD_PIN3,GPIO.LOW)
    time.sleep(1)

# 7
    GPIO.output(BCD_PIN0,GPIO.HIGH)
    GPIO.output(BCD_PIN1,GPIO.HIGH)
    GPIO.output(BCD_PIN2,GPIO.HIGH)
    GPIO.output(BCD_PIN3,GPIO.LOW)
    time.sleep(1)

# 8
    GPIO.output(BCD_PIN0,GPIO.LOW)
    GPIO.output(BCD_PIN1,GPIO.LOW)
    GPIO.output(BCD_PIN2,GPIO.LOW)
    GPIO.output(BCD_PIN3,GPIO.HIGH)
    time.sleep(1)

# 9
    GPIO.output(BCD_PIN0,GPIO.HIGH)
    GPIO.output(BCD_PIN1,GPIO.LOW)
    GPIO.output(BCD_PIN2,GPIO.LOW)
    GPIO.output(BCD_PIN3,GPIO.HIGH)
    time.sleep(1)
```



RFID (EM18)

Radio frequency Identification i.e. RFID is a wireless identification technology that uses radio waves to identify the presence of RFID tags.

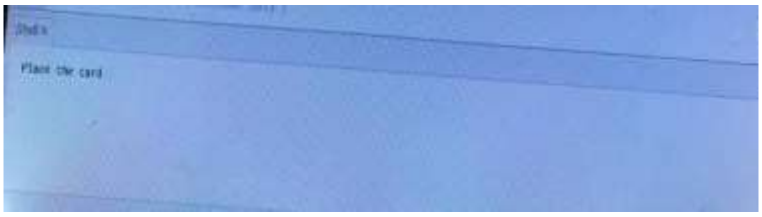
Just like Bar code reader, RFID technology is used for identification of people, object etc. presence.

In barcode technology, we need to optically scan the barcode by keeping it in front of reader, whereas in RFID technology we just need to bring RFID tags in range of readers. Also, barcodes can get damaged or unreadable, which is not in the case for most of the RFID.

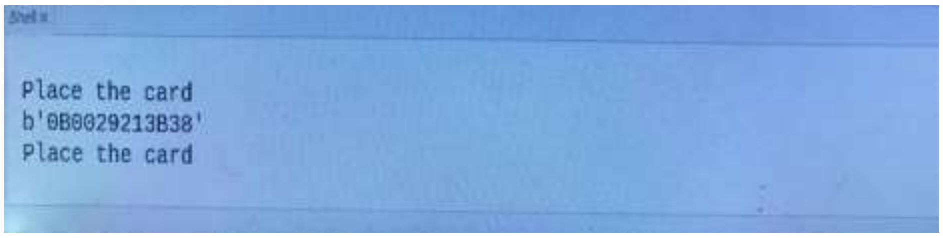
RFID is used in many applications like attendance system in which every person will have their separate RFID tag which will help identify person and their attendance. RFID is used in many companies to provide access to their authorized employees. It is also helpful to keep track of goods and in automated toll collection system on highway by embedding Tag (having unique ID) on them.

Code

```
# Setting up Raspberry Pi for Serial Communication BEFORE TO RUN PROGRAM
# CONNECT TX of RFID to BOARD of 10 PIN OR RX OF RASPBERRY PI BOARD
import time
import serial

data = serial.Serial(
            port='/dev/ttyS0',
            baudrate = 9600,
            parity=serial.PARITY_NONE,
            stopbits=serial.STOPBITS_ONE,
            bytesize=serial.EIGHTBITS
            )
            #timeout=1 # must use when using data.readline()
            #)
print (" ")
try:
  while 1:
     #x=data.readline()#print the whole data at once
     #x=data.read()#print single data at once

     print ("Place the card")
     x=data.read(12)#print upto 10 data at once and the
     print (x)

except KeyboardInterrupt:
    data.close()
```



```
Shell ×

Place the card
```



```
Shell ×

Place the card
b'0B0029213B38'
Place the card
```