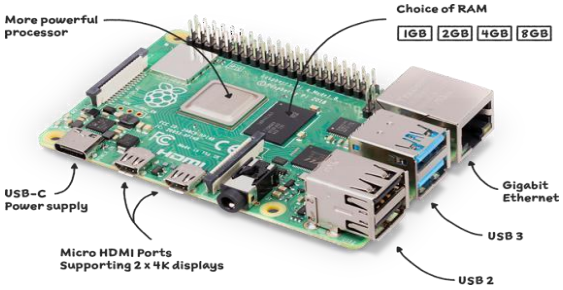
PRACTICAL NO 1

Aim: Raspberry Pi Hardware Preparation and Installation

Hardware Guide :

For getting started with raspberry pi for the first time you will require the following hardware

1. **Raspberry Pi (latest Model):**



2. **Monitor or TV:**

3. **HDMI cable:**

4. **Ethernet cable:**

5. **USB keyboard:**

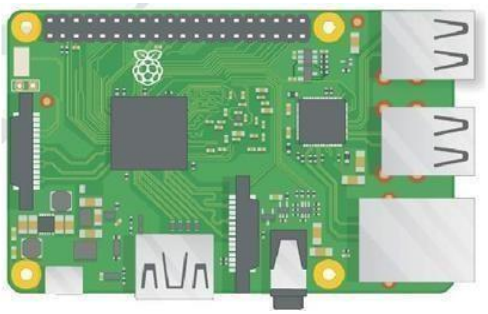
6. **USB mouse:**

7. **Micro USB power supply:**

8. **8GB or larger microSD card:**

9. **SD Card Reader:**

Raspberry Pi 3 Model B:



The Raspberry Pi 3 is the third generation Raspberry Pi. It replaced the Raspberry Pi 2 Model B in February 2016. Compared to the Raspberry Pi 2 it has:

A 1.2GHz 64-bit quad-core ARMv8 CPU 802.11n Wireless LAN

Bluetooth 4.1

Bluetooth Low Energy (BLE) Like the Pi 2, it also has:

4 USB ports 40 GPIO pins Full HDMI port Ethernet port

Combined 3.5mm audio jack and composite video Camera interface (CSI) Display interface (DSI)

Micro SD card slot (now push-pull rather than push-push) Video Core IV 3D graphics core

The Raspberry Pi 3 has an identical form factor to the previous Pi 2 (and Pi 1 Model B+) and has complete compatibility with Raspberry Pi 1 and 2.

Monitor or TV:

A monitor or TV with HDMI in can be used as a display with a Raspberry Pi. Most modern television sets and monitors have an HDMI port, and are the easiest to get working with the Raspberry Pi. You can use an HDMI cable to connect the Raspberry Pi directly to the t elevision or monitor.

Some older monitors have a DVI port. These work well with the Raspberry Pi, although you'll need an HDMI-to-DVI adapter to attach to an HDMI cable, or a one-piece HDMI-to-DVI cable. Some old monitors have a VGA port. These can be trickier to use as you'll need an HDMI-to-VGA converter, which can change digital video to analogue video. A simple port adapter won't work.

HDMI to HDMI Cable:

Connect Raspberry Pi to a Monitor or TV with a HDMI to HDMI cable.

Ethernet cable:

Ethernet cable will allow your Pi to connect with the internet. It is also useful for headless setup of Raspberry Pi

USB Keyboard and Mouse:

Any standard USB keyboard and mouse can be used with the Raspberry Pi. This plug and play devices will work without any additional driver. Simply plug them into the Raspberry Pi and they should be recognised when it starts up.

Power Supply:

It is recommended that you use a 5V, 2A USB power supply for all models of Raspberry Pi.

SD Card:

The latest version of Raspbian, the default operating system recommended for the Raspberry Pi, requires an 8GB (or larger) micro SD card. SD card will store the operating systems as well as all the file and applications created by you.

Installation Guide:

Now since you have all the required hardware, we will now learn how to get the operating system onto your microSD card so that you can start using software on your Raspberry Pi

Get Raspbian OS on your microSD card:

Raspbian comes pre-installed with plenty of software for education, programming and general use. It has Python, Scratch, Sonic Pi, Java, Mathematica and more.

1. To download Raspbian log on to raspberrpi.org and click on the download, then click on Raspbian and lastly download the RASPBIAN BUSTER WITH DESKTOP file. You can choose either the Torrent file or ZIP file.
The downloaded file will be in zip format. To unzip the file, you will require an unzip tool. You can use any unzipping tool viz. WINRAR, 7ZIP etc. After unzipping the file, you will find a disc image file in the unzipped folder.

2. Now format the SD Card before writing the disc image file on the SD card. You can use SD Formatter tool or any other tool of your wish.

3. To write the image file of the operating system on the SD card you will require a Disk Imager tool. For this you can use Win32 Disk Imager tool.

4. Once the image is written on the SD Card, your untitled SD card will now have the name boot. Your SD Card will now hold the Raspbian Operating system required for the first-time setup.

Plugging in your Raspberry Pi:

1. Begin by placing your SD card into the SD card slot on the Raspberry Pi. It will only fit one way.

2. Next, plug your keyboard and mouse into the USB ports on the Raspberry Pi.

3. Make sure that your monitor or TV is turned on, and that you have selected the right in put (e.g. HDMI 1, DVI, etc).

4. Connect your HDMI cable from your Raspberry Pi to your monitor or TV.

5. If you intend to connect your Raspberry Pi to the internet, plug an Ethernet cable into the Ethernet port, or connect a WiFi dongle to one of the USB ports (unless you have a Raspberry Pi 3).

6. When you're happy that you have plugged all the cables and SD card in correctly, connect the micro USB power supply. This action will turn on and boot your Raspberry Pi.

NOTE: for Raspberry Pi 4 please follow the instruction given below.

After writing the image on SD Card plug in the SD Card in to the Raspberry pi 4. If boot normally and you see the raspberry pi screen than ok. But if not than make the following changes in the SD card ‘config' file with notepad++

1. Uncomment the line 'hdmi_force_hotplug=1’
2. Uncomment the line 'hdmi_group=1’
3. Uncomment the line 'hdmi_mode=1'
4. Save and exit the 'config' file
5. Now insert again the memory card into the raspberry pi it will work.

PRACTICAL NO 2

Aim: GPIO: Light the LED With python

Hardware Guide

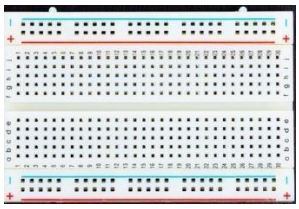
Along with the basic setup you will require the following components to get started with the GPIO pins as follows:

1. **LED**

2. **Resistor**

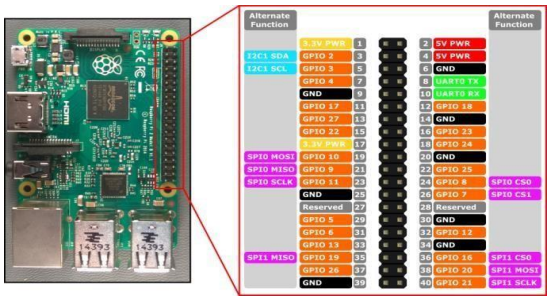
3. **Connecting wires**

4. **Breadboard**



Before learning this lesson, you must understand the pin numbering system of the GPIO pins

GPIO?



GPIO, or General Purpose Input/Output, refers to digital signal pins on an integrated circuit or electronic board that can be configured as either inputs or outputs, and are controlled by software. They serve as a flexible interface between a microcontroller and external devices.

GPIO (General Purpose Input/Output) pins are digital signal pins that can be controlled via software. They can be used for:

- **Input**: Reading the state of a button, sensor, etc.
- **Output**: Controlling an LED, relay, motor, etc.

One powerful feature of the Raspberry pi is the row of GPIO pins along the top edge of the board

What are they for? What can we do with them?

GPIO (General Purpose Input/Output) pins are tiny connectors on devices like **Raspberry Pi or microcontrollers** that let them **talk to or control other electronics**.

You can program the pins to interact in amazing ways with the real world. Inputs don't have to come from a physical switch; it could be input from a sensor or a signal from another computer or device

- **Turn things on/off** (like lights or motors)
- **Read buttons or sensors** (like temperature or motion)

Build smart devices (like a home automation system) In short: **GPIO lets your device interact with the real world**

- **How the GPIO pins work?**

  GPIO pins (General Purpose Input/Output) are digital signal pins on microcontrollers that can be configured as either inputs or outputs, allowing the microcontroller to interact with the external world. As inputs, they can read signals from switches, sensors, etc. As outputs, they can drive LEDs, motors, and other components.

1. Input or Output Mode

Each GPIO pin can be set to work in one of two modes:

- **Input** – it **reads** signals from things like buttons or sensors.
- **Output** – it **sends** signals to control things like LEDs, motors, or relays.

2. HIGH or LOW (1 or 0)

GPIO pins use **electric signals**:

- **HIGH (1)** = voltage is on (usually 3.3V or 5V)
- **LOW (0)** = voltage is off (0V)

In **Output mode**, the pin sends HIGH or LOW to control devices

In **Input mode**, it checks if it's receiving HIGH or LOW from a device.

Example Use

Press a **button** → GPIO pin reads it as **input**

Turn on an **LED** → GPIO pin sends **output**

In short:

GPIO pins act like switches or sensors that you can control with code.
Warning: Randomly plugging wires and power sources into your pi, however, may kill it. Bad things can also happen if you try to connect things to your pi that use a lot of power.

A note on pin numbering:

There are usually **two ways to number GPIO pins** on boards like the Raspberry Pi:

1. **Physical Pin Numbering**
   - Counts pins by their position on the board (1, 2, 3, 4, …)
   - Easy to find on the board itself
   - Example: Pin 1 is the top-left pin
2. **GPIO (Broadcom) Numbering**
   - Uses the chip's internal GPIO numbers (GPIO17, GPIO18, etc.)
   - Matches the software library's naming
   - Example: Physical Pin 12 is GPIO18

Code :

```
#Blink LED Program
#Connect the LED to GPIO 22 Pin
#LED Blink Program
#Connect the LED to GPI022(i.e. Physical pin 15)
#import GPIO and time library
import RPi.GPIO as GPIO
import time import sleep
GPIO.setmode(GPIO.BCM)                    #set the Pin mode you will be working with
ledPin =22                                #this is GPI022 pin i.e Physical Pin 15
                                          #setup is the ledPin(i.e.GPIO22)as output
GPIO.setup(ledPin, GPIO.OUT)
GPIO.output(ledPin, False)
try;
while True:
GPIO.output(ledPin, True)                 #Set the LED Pin to HIGH
print("LED ON")
sleep(1) #Wait for 1 sec
GPIO.output(ledPin, False)                #Set the LED Pin to LOW
print("LED OFF")
sleep(1) #Wait for 1 sec
finally:
                                          #reset the Gpio Pins
GPIO.output(ledPin, False)
GPIO.cleanup()
#end of code
```

PRACTICAL NO 3

Aim: Displaying different LED patterns with Raspberry pi.

For displaying different LED pattern connecte 8 LEDs in the same format to the pin number given in the below python code:

Hardware Guide:

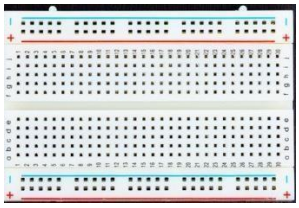
Along with the basic setup you will require the following components to get started with the GPIO pins as follows

1. **LED**

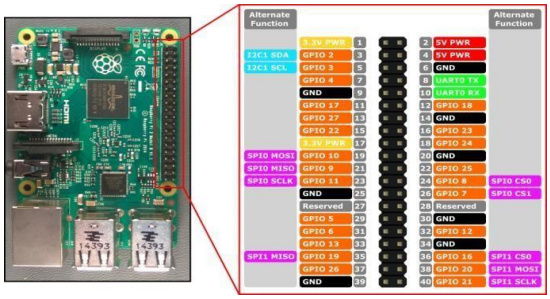
2. **Resistor**

3. **Connecting wires**

4. **Breadboard**



GPIO?



code for LED pattern:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
```

```
led1 = 29
led2 = 31
led3 = 33
led4 = 35
led5 = 36
led6 = 37
led7 = 38
led8 = 40
# setup the ledPin (i.e. GPIO22) as output
GPIO.setup(led1, GPIO.OUT)
GPIO.setup(led2, GPIO.OUT)
GPIO.setup(led3, GPIO.OUT)
GPIO.setup(led4, GPIO.OUT)
GPIO.setup(led5, GPIO.OUT)
GPIO.setup(led6, GPIO.OUT)
GPIO.setup(led7, GPIO.OUT)
GPIO.setup(led8, GPIO.OUT)
GPIO.output(led1, False)
GPIO.output(led2, False)
GPIO.output(led3, False)
GPIO.output(led4, False)
GPIO.output(led5, False)
GPIO.output(led6, False)
GPIO.output(led7, False)
GPIO.output(led8, False)
def ledpattern(ledVal1, ledVal2, ledVal3, ledVal4, ledVal5, ledVal6, ledVal7, ledVal8):
   GPIO.output(led1, ledVal1)
   GPIO.output(led2, ledVal2)
   GPIO.output(led3, ledVal3)
   GPIO.output(led4, ledVal4)
   GPIO.output(led5, ledVal5)
   GPIO.output(led6, ledVal6)
   GPIO.output(led7, ledVal7)
   GPIO.output(led8, ledVal8)
def patternOne():
```

```
  for i in range(0, 3):
    ledpattern(1, 0, 1, 0, 1, 0, 1, 0)
    time.sleep(1)
    ledpattern(0, 1, 0, 1, 0, 1, 0, 1)
    time.sleep(1)
def patternTwo():
    for i in range(0, 5):
        ledpattern(1, 0, 0, 0, 0, 0, 0, 0)
        time.sleep(0.1)
        ledpattern(0, 1, 0, 0, 0, 0, 0, 0)
        time.sleep(0.1)
        ledpattern(0, 0, 1, 0, 0, 0, 0, 0)
        time.sleep(0.1)
        ledpattern(0, 0, 0, 1, 0, 0, 0, 0)
        time.sleep(0.1)
        ledpattern(0, 0, 0, 0, 1, 0, 0, 0)
        time.sleep(0.1)
        ledpattern(0, 0, 0, 0, 0, 1, 0, 0)
        time.sleep(0.1)
        ledpattern(0, 0, 0, 0, 0, 0, 1, 0)
        time.sleep(0.1)
        ledpattern(0, 0, 0, 0, 0, 0, 0, 1)
        time.sleep(0.1)
def patternThree():
    for i in range(0, 5):
        ledpattern(0, 0, 0, 0, 0, 0, 0, 1)
        time.sleep(0.1)
        ledpattern(0, 0, 0, 0, 0, 0, 1, 0)
        time.sleep(0.1)
        ledpattern(0, 0, 0, 0, 0, 1, 0, 0)
        time.sleep(0.1)
        ledpattern(0, 0, 0, 0, 1, 0, 0, 0)
        time.sleep(0.1)
        ledpattern(0, 0, 0, 1, 0, 0, 0, 0)
        time.sleep(0.1)
```

```
        ledpattern(0, 0, 1, 0, 0, 0, 0, 0)
        time.sleep(0.1)
        ledpattern(0, 1, 0, 0, 0, 0, 0, 0)
        time.sleep(0.1)
        ledpattern(1, 0, 0, 0, 0, 0, 0, 0)
        time.sleep(0.1)
def patternFour():
    for i in range(0, 5):
        ledpattern(0, 1, 1, 1, 1, 1, 1, 1)
        time.sleep(0.1)
        ledpattern(1, 0, 1, 1, 1, 1, 1, 1)
        time.sleep(0.1)
        ledpattern(1, 1, 0, 1, 1, 1, 1, 1)
        time.sleep(0.1)
        ledpattern(1, 1, 1, 0, 1, 1, 1, 1)
        time.sleep(0.1)
        ledpattern(1, 1, 1, 1, 0, 1, 1, 1)
        time.sleep(0.1)
        ledpattern(1, 1, 1, 1, 1, 0, 1, 1)
        time.sleep(0.1)
        ledpattern(1, 1, 1, 1, 1, 1, 0, 1)
       time.sleep(0.1)
       ledpattern(1, 1, 1, 1, 1, 1, 1, 0)
       time.sleep(0.1)
def patternFive():
    for i in range(0, 5):
        ledpattern(1, 1, 1, 1, 1, 1, 1, 0)
        time.sleep(0.1)
        ledpattern(1, 1, 1, 1, 1, 1, 0, 1)
        time.sleep(0.1)
        ledpattern(1, 1, 1, 1, 1, 0, 1, 1)
        time.sleep(0.1)
        ledpattern(1, 1, 1, 1, 0, 1, 1, 1)
        time.sleep(0.1)
        ledpattern(1, 1, 1, 0, 1, 1, 1, 1)
```

```
        time.sleep(0.1)
        ledpattern(1, 1, 0, 1, 1, 1, 1, 1)
        time.sleep(0.1)
        ledpattern(1, 0, 1, 1, 1, 1, 1, 1)
        time.sleep(0.1)
        ledpattern(0, 1, 1, 1, 1, 1, 1, 1)
        time.sleep(0.1)
try:
    while True:
        patterOne()
        patternTwo()
        patternThree()
        patternFour()
        patternFive()
finally:
    # reset the GPIO Pins
    GPIO.cleanup()
```

PRACTICAL NO 4

Aim: Displaying Time over 4 Digit 7 Segment Display using Raspberry Pi.

Hardware Guide:

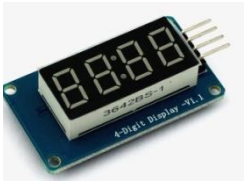
For completing this lesson ,you will require the following things along with your initial raspberry pi. Setup.

1. TM1637 4-digit seven segment Display board
2. Connecting wires

TM1637 4-digit seven segment Display board:

This is a common anode 4-digit tube display module which uses the TM1637 driver chip; Only 2 connections are required to control the 4-digit 8-segment displays

Here is the module

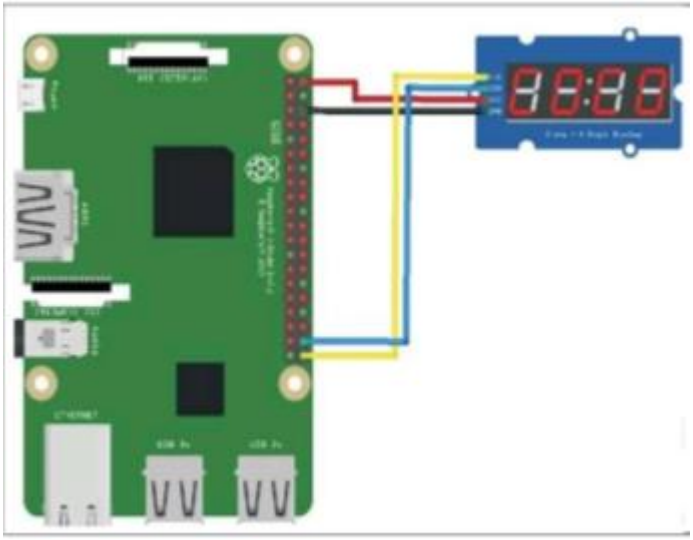


Features of the module

1. Display common anode for the four red LED.
2. Powered supply by 3.3V/5V.
3. Four common anode tube display module is driven by IC TM1637.

Wiring up your Circuit:

1. Connect the Pin2 (5V) of Rpi to Vcc pin of Module
2. Connect Pin 6 (GND) of Rpi to GND of Module
3. Connect Pin38 (GPIO20) of Rpi to DIO of Module
4. Lastly connect Pin 40 (GPIO21) of Rpi to CLK of Module.



Software Guide:

1. Now to download libraries, open Web Browser on your Raspberry Pi and log on to the Following link: https://github.com/timwaizenegger/raspberrypi examples/tree/master/actor-led-7segment-4numbers. Click on the actor-led-7segment 4numbers.zip folder and Now click on Download Button to download the file.
2. Now on your rpi move to /home/pi/Downloads/ location to find the zip file downloaded.
3. Unzip the file and try to execute the different example codes present in that folder in Python 2 Idle.
4. Now open Python 2 Idle, create a new file, write the code given below and save it in the same folder i.e. actor-led-7segment-4numbers since the code below is depended on tm1637.py file which is present in the same folder.

Code:

```
#Program to display TIme on 4-digit Seven segment display
from time import sleep
import tm1637
try:
import thread
except importError:
import_thread as thread
# initialize the clock(GND, VCC=3,3V,Example Pins are DIO-20 and CLK21)
Display = tm1637.TM1637(CLK=21,DIO=20,brightness=1.0)
try:
print("Starting clock in the backgroung(press CTRL+C to stop):")
Display.Starting Clock(military_time=True)
Display.SetBrightness(1.0)
while True:
Display.ShowDoublepoint(True)
sleep(1)
Display.ShowDoublepoint(False)
sleep(1)
Display.StopClock()
thread.interrupt_main();
except Keyboardinterrupt:
print("Properly closing the clock and openGPIO pins")
Display.cleanup()
```

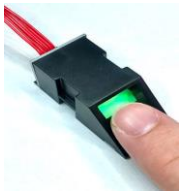
PRACTICAL NO 5

Aim: Fingerprint Sensor interfacing with Raspberry Pi.

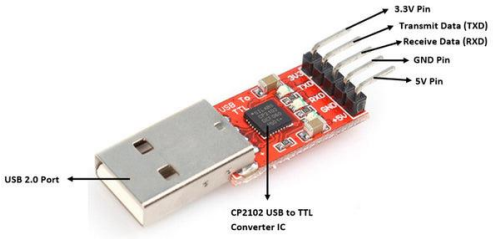
Hardware Guide:

For completing this lesson, you will require the following things along with your initial raspberry pi. Setup

1. **Fingerprint Sensor**



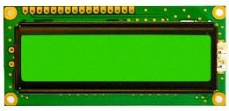
2. **USB to TTL/UART converter**



3. **Connecting wires**

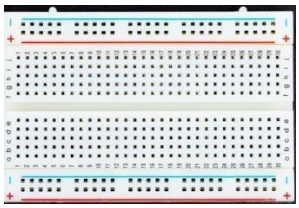
4. **Push Buttons**

5. **16x2 LCD**

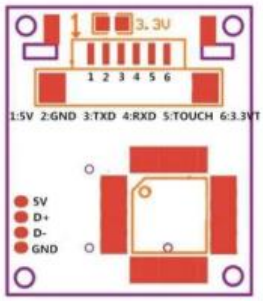


6. **LED**

7. **Breadboard**



Fingerprint Sensor:
It is an intelligent module which can freely get fingerprint, image processing, verified fingerprint, search and storage, and it can work normally without upper monitor's participatory management. Fingerprint processing includes two parts: fingerprint enrolment and fingerprint matching(the matching can be 1:1 or 1: N). Enrolling fingerprint, user needs to enter the finger 2-4 times for every one finger, process finger images with many times, store generate templates on module. When fingerprint matching, enrol and process verified fingerprint image and then matching with
Pin 1 +5V, Pin2 = GND, PIN 3 = TXD, Pin 4 = RXD



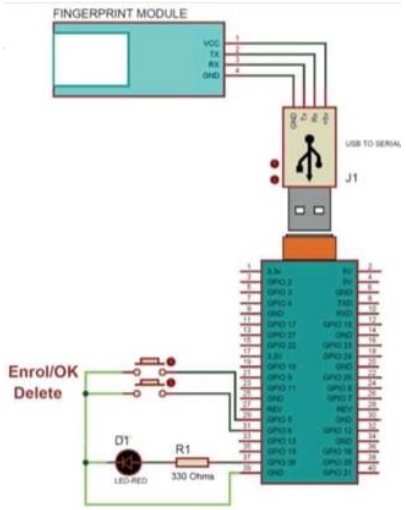
module (if match with appoint templates on the module, named fingerprint verification, for 1:1 matching method; if match with many templates on the module, named fingerprint search method also named 1: N) system will return the matching result, success or failure.

USB to TTL converter:
Here we have used a fingerprint module which works on UART. So here we have interfaced this fingerprint module with Raspberry Pi using a USB to Serial converter.

Wiring up your circuit:
In this Raspberry Pi Finger Print sensor interfacing project, we have used a 2 push buttons:
one for enrolling the new finger print, one for deleting the already fed finger prints positions. A LED is used for indication that fingerprint sensor is ready to take finger for matching. Here we have used a fingerprint module which works on UART. So here we have interfaced this fingerprint module with Raspberry Pi using a USB to Serial converter. So, first of all, we need to make the all the required connection as shown below. Connections are simple, we have just connected fingerprint module to Raspberry Pi USB port by using USB to Serial converter.

1. Connect the VCC Pin of Finger Print Module (Pin 1) Red Wire to 5V Pin of USB to TTL converter
2. Connect the GND Pin of Finger Print Module (Pin 2) Black Wire to GND Pin of USB to TTL converter.

3. Connect the TXD Pin of Finger Print Module (Pin 3) Yellow Wire to RXD Pin of USB to TTL converter
4. Connect the RXD Pin of Finger Print Module (Pin 4) White Wire to TXD Pin of USB to TTL converter.
5. Lastly connect the USB to TTL converter to USB port of Raspberry Pi.
6. Connect one push button switch to pin 29 of Raspberry Pi for Enroll finger print.
7. Connect another push button switch to pin 31 of Raspberry Pi for Deleting already fed finger print.
8. Connect LED to pin 37 of Raspberry Pi for indication that finger print is ready to take finger for matching.
9. Finally connect ground pin 39(GND) of Raspberry pi to circuit as shown in Circuit diagram.



Software Guide:

After making all the connections we need to power up Raspberry Pi and get it ready with terminal open. Now we need to install fingerprint library for Raspberry Pi in python language by following the below steps.

To install this library, root privileges are required. So first we enter in root by given

1. command: sudo bash
2. Then download some required packages by using given commands:
   a. wget-0-http://apt.pm-codeworks.de/pm-codeworks.de.gpg | apt-key add –
   b. wget http://apt.pm-codeworks.de/pm-codeworks.list -P /etc/apt/sources.list.d/
3. After this, we need to update the Raspberry pi and install the downloaded finger print
4. sensor library:
   a. sudo apt-get update
   b. sudo apt-get install python-fingerprint
   c. now exit root by typing exit

5. After installing library now, we need to check USB port on which your finger print sensor is connected, by using given the command: Is /dev/ttyUSB* (or Isusb)

Now in your python code replace the USB port number with the one you got on the screen after executing the command in step4.

Code:

```
import time
from pyfingerprint.pyfingerprint import PyFingerprint
import RPI.GPIO as gpio
enrol=5
delet=6
led=26
HIGH=1
LOW=0
gpio.setwarnings (False)
gpio.setmode(gpio.BCM)
gpio.setup(enrol, gpio.IN, pull_up_down=gpio.PUD_UP)
gpio.setup(delet, gpio.IN, pull_up_down=gpio.PUD_UP)
gpio.setup(led, gpio.OUT)
try:
f = PyFingerprint('/dev/ttyUSB0', 57600, 0xFFFFFFFF, 0x00000000)
if (f.verifyPassword() == False ):
raise ValueError('The given fingerprint sensor password is wrong!")
except Exception as e:
print('Exception message: ' + str(e))
exit(1)
print('Currently used finger templates: ' + str(f.getTemplateCount()) +'/'+
str(f.getStorage Capacity()))
def enrollFinger():
print('Waiting for finger...')
while (f.readImage() == False):
pass
f.convertImage(0x01)
result = f.searchTemplate()
position Number = result[0]
if (position Number >= 0):
print('Finger already exists at position #' + str(position Number))
time.sleep(2)
return
print('Remove finger...')
time.sleep(2)
print('Waiting for same finger again...')
while (f.readImage() == False ):
pass
f.convertImage(0x02)
if (f.compareCharacteristics() == 0):
print ("Fingers do not match")
```

```
time.sleep(2)
f.createTemplate()
return
position Number = f.storeTemplate()
print('Finger enrolled successfully!')
print('New template position #' + str(position Number))
time.sleep(2)
def searchFinger():
try:
print('Waiting for finger...')
while(f.readImage() == False):
#pass
time.sleep(.5)
return
f.convertimage(0x01)
result = f.searchTemplate()
try:
print('Waiting for finger...')
while(f.readImage() == False):
#pass
time.sleep(.5)
return
f.convertImage(0x01)
result = f.searchTemplate()
position Number = result[0]
accuracyScore = result[1]
if position Number == -1 :
print('No match found!')
time.sleep(2)
return
else:
print('Found finger at position #' + str(positionNumber))
time.sleep(2)
except Exception as e:
print('Operation failed!")
print('Exception message: ' + str(e))
exit(1)
def delete Finger():
positionNumber = 0
count=0
while gpio.input(delet) == True: # here delet key means ok
positionNumber = input('Please enter the template position you want to delete: ')
position Number = int(positionNumber)
if f.deleteTemplate (position Number) == True :
print('Template deleted!")
time.sleep(1)
print('Currently used finger templates: ' + str(f.getTemplateCount()) +'/'+
```

```
str(f.getStorageCapacity()))
time.sleep(1)
return
print ("Edkits Electronics Welcomes You ")
time.sleep(3)
flag=0
while 1:
gpio.output(led, HIGH)
if gpio.input(enrol) == 0:
gpio.output(led, LOW)
enrollFinger()
elif gpio.input(delet) == 0:
gpio.output(led, LOW)
while gpio.input(delet) == 0:
time.sleep(0.1)
deleteFinger()
else:
searchFinger()
```

PRACTICAL NO 6

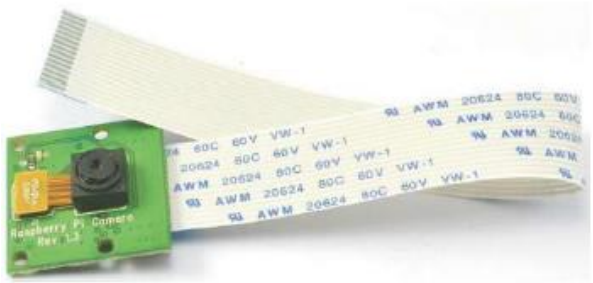
Aim: Capturing Images with Raspberry Pi and Pi Camera.

Hardware Guide:

For completing this lesson, you will require the Camera Module along with your initial raspberry pi setup.

Camera Module:

The Raspberry Pi Camera Board plugs directly into the CSI connector on the Raspberry Pi. The camera is supported in the latest version of Raspbian, the Raspberry Pi's preferred operating system.



The Raspberry Pi Camera Board Features:

1. Fully Compatible with Both the Model A and Model B Raspberry Pi
2. 5MP Omnivision 5647 Camera Module
3. Still Picture Resolution: 2592 x 1944
4. Video: Supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 Recording
5. 15-pin MIPI Camera Serial Interface - Plugs Directly into the Raspberry Pi Board
6. Size: 20 x 25 x 9mm
7. Weight 3g
8. Fully Compatible with many Raspberry Pi cases

Connect the Camera Module:

First of all, with the Pi switched off, you'll need to connect the Came ra Module to the Raspberry Pi's camera port, then start up the Pi and ensure the software is enabled.

1. Locate the camera port and connect the camera
2. Start up the Pi.
3. Open the Raspberry Pi Configuration Tool from the main menu.
4. Ensure the camera software is enabled. If it's not enabled, enable it and reboot your Pi to begin.

Software Guide:

Now your camera is connected and the software is enabled, you can get started by capturing an image. You can capture an image by just typing a single line c ommand.
Open terminal window and type the

command as follows:

$ sudo raspistill-o/home/pi/Desktop/image.jpg

This command will capture an image and store it at the specified location (here the location specified is /home/pi/Desktop) with the specified name (here the name is 'image.jpg').

You can even write a code in Python to capture an image using raspberry pi camera. Open Python3, create a new file and type the code as follows:

Code:

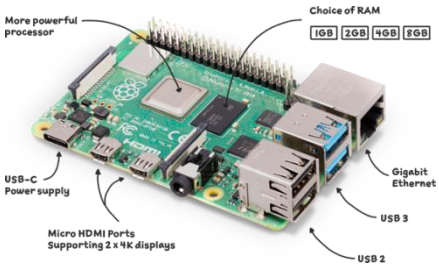
```
#Camera Program
#import time and picamera library
from time import sleep
from picamera import PiCamera
camera = PiCamera()
camera.resolution = (1280, 720) # selecting resolution 1280x720 px
camera.start_preview()
#Camera warm-up time
sleep(2)
camera.capture('/home/pi/Pictures/newimage.jpg') #capture and save image at specified location
camera.stop_preview()
```

PRACTICAL NO 7

Aim: Interfacing Raspberry Pi With RFID RC522 Chip.

Hardware Guide:

1. **Raspberry Pi.**

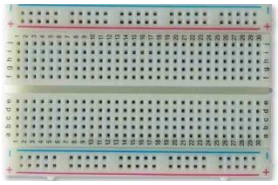


2. **Micro SD Card**

3. **Power Supply**

4. **RC522 RFID Reader**

5. **Breadboard**



6. **Breadboard wire**

Wiring the RFID RC522

On your RFID RC522 you will notice that there are 8 possible connections on it, these being SDA (Serial Data Signal), SCK (Serial Clock), MOSI (Master Out Slave In), MISO (Master In Slave Out), IRQ (Interrupt Request), GND (Ground Power), RST (Reset Circuit) and 3.3v (3.3v Power In). We will need to wire all of these but the IRQ to our Raspberry Pi's GPIO pins.

You can either wire these directly to the GPIO Pins or like we did in this tutorial, plug the RFID RC522 into our Breadboard then wire from there to our Raspberry Pi's GPIO Pins. Wiring your RFID RC522 to your Raspberry Pi is fairly simple, with it requiring you to connect just 7 of the GPIO Pins directly to the RFID reader. Follow the table below, and check out our GPIO guide to see the positions of the GPIO pins that you need to connect your RC522 to.

- **SDA connects to Pin 24.**
- **SCK connects to Pin 23.**
- **MOSI connects to Pin 19.**
- **MISO connects to Pin 21.**
- **GND connects to Pin 6.**
- **RST connects to Pin 22.**
- **3.3v connects to Pin 1.**

Setting up Raspbian for the RFID RC522:

Before we begin the process of utilizing the RFID RC522 on our Raspberry Pi, we will first have to make changes to its configuration. By default, the Raspberry Pi has the SPI (Serial Peripheral Interface) disabled, which is a bit of a problem as that is what our RFID reader circuit runs through.

follow our steps below to configure your Raspberry Pi and Raspbian to utilize the SPI interface.

1. Let's begin by first opening the raspi-config tool, and we can do this by opening the terminal and running the following command. "sudo raspi-config"
2. This tool will load up a screen showing a variety of different options. On here use the arrow keys to select "5 Interfacing Options". Once you have this option selected, press Enter.
3. Now on this next screen, you want to use your arrow keys to select "P4 SPI", again press Enter to select the option once it is highlighted.
4. You will now be asked if you want to enable the SPI Interface, select Yes with your arrow keys and press Enter to proceed. You will need to wait a little bit while the raspi-config tool does its thing in enabling SPI.
5. Once the SPI interface has been successfully enabled by the raspi-config tool you should see the following text appear on the screen, "The SPI interface is enabled ". Before the SPI Interface is fully enabled, we will first have to restart the Raspberry Pi. To do this first get back to the terminal by pressing Enter and then ESC.
   Type the following Linux command into the terminal on your Raspberry Pi to restart your Raspberry Pi.
   "sudo reboot"
6. Once your Raspberry Pi has finished rebooting, we can now check to make sure that it has in fact been enabled. The easiest way to do this is to run the following command to see if spi bcm2835 is listed. "Ismod | grep spi"
   If you see spi_bcm2835, then you can proceed on with this tutorial and skip on to the next section. If for some reason it had not appeared when you entered the previous command, try following the next three steps.

7. If for some reason the SPI module has not activated, we can edit the boot configuration file manually by running the following command on our Raspberry Pi.
   "sudo nano/boot/config.txt"
8. Within the configuration file, use CTRL + W to find "dtparam=spi=on".
   If you have found it, check to see if there is a # in front of it. If there is, remove it as this is commenting out the activation line. If you can't find the line at all, add "dtparam=spi=on" to the bottom of the file.
   Once you have made the changes, you can press CTRL + X then pressing Y and then Enter to save the changes.
   You can now proceed from Step 5 again, rebooting your Raspberry Pi then checking to see if the module has been enabled.

Getting Python ready for the RFID RC522

Now that we have wired up our RFID RC522 circuit to the Raspberry Pi we can now power it on and begin the process of programming simple scripts in Python to interact with the chip. The scripts that we will be showing you how to write will show you how to read data from the RFID chips and how to write to them. These will give you the basic idea of how data is dealt with and will be the basis of further RFID RC522 tutorials.

1. Before we start programming, we first need to update our Raspberry Pi to ensure it's running the latest version of all the software. Run the following two commands on your Raspberry Pi to update it.
   "sudo apt update"
   "sudo apt upgrade"
2. Now the final thing we need before we can proceed is to install **python3-dev, python-pip and git packages**. Simply run the following command on your Raspberry Pi to install all of the required packages for this guide on setting up your RFID reader.
   "sudo apt install python3-dev python3-pip"
3. To begin, we must first install the Python Library spidev to our Raspberry Pi using the python "pip" tool that we downloaded in the previous step.

   The spidev library helps handle interactions with the SPI and is a key component to this tutorial as we need it for the Raspberry Pi to interact with the RFID RC522. Run the following command on your Raspberry Pi to install spidev to your Raspberry Pi through pip.
   Sudo is used here to ensure that the package is installed so that all users
   can utilize it and not just the current user.
   sudo pip3 install spidev
4. Now that we have installed the spidev library to our Raspberry Pi we can now now proceed to installing the MFRC522 library using pip as well. You can check out the RFID MFRC522 Python code from the PiMyLifeUp Github if you are interested in seeing how the library works or improving its behaviour.

   There are two files that are included within our MFRC522 library that we make use of:
   - **MFRC522.py** which is an implementation of the RFID RC522 interface, this library handles all the heavy lifting for talking with the RFID over the Pi's SPI Interface.

- **SimpleMFRC522.py** that takes the MFRC522.py file and greatly simplifies it by making you only have to deal with a couple of functions instead of several.

To install the MFRC522 library to your Raspberry Pi using pip go ahead and run the following command.
"sudo pip3 install mfrc522"

5. With the library now saved to our Raspberry Pi, we can begin programming for our RFID RC522. To start off with we will be showing you how to write data to your RFID cards by using the RC522. Simply go onto our next section to begin programming our first Python script.

Writing with the RFID RC522

For our first Python script, we will be showing you how to write data from the RC522 to your RFID tags. Thanks to the SimpleMFRC522 script this will be relatively simple, but we will still go into how each part of the code works.

1. Now lets start off by making a folder where we will be storing our couple of scripts. We will be calling this folder "pi-rfid", create it by running the following command. "mkdir ~/pi-rfid"
2. Begin by changing directory into our newly cloned folder, and begin writing our Write.py Python script. "cd ~/pi-rfid"

sudo nano Write.py

3. **Code:**

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
from mfrc522 import SimpleMFRC522
reader = SimpleMFRC522()
try:
text = input('New data:')
print("Now place your tag to write")
reader.write(text)
print("Written")
finally:
GPIO.cleanup()
```

4. Once you are happy that the code looks correct, you can save the file by pressing CTRL + X then pressing Y and then finally hitting ENTER.
5. Now that we have written our script, we will want to test it out. Before testing out the script make sure that you have an RFID tag handy. Once ready, type the following command into your Raspberry Pi's terminal. "sudo python3 Write.py"
6. You will be asked to write in the new data, in our case we are going to just type in as its short and simple. Press Enter when you are happy with what you have written.
7. With that done, simply place your RFID Tag on top of your RFID RC522 circuit. As soon as it detects it, it will immediately write the new data to the tag. You should see "Written" appear in your command line if it was successful.

8. You can look at our example output below to see what a successful run looks like.
   pi@raspberrypi:~/pi-rfid $ sudo python3 Write.py
9. Now place your tag to write
10. Written

You have now successfully written your Write.py script, and we can now proceed to

1. show you how to read data from the RFID RC522 in the next segment of this tutorial.
2. Reading with the RFID RC522
3. Now that we have written our script to write to RFID tags using our RC522 we can now write a script that will read this data back off the tag.
4. Let's start off by changing the directory to make sure we are in the right place, and then we can run nano to begin writing our Read.py script. "cd ~/pi-rfid"
5. sudo nano Read.py
6. Within this file, write the following lines of code. This script will basically sit and wait till you put your RFID tag on the RFID RC522 reader, it will then output the data it reads off the tag
7. **Code.**

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
from mfrc522 import SimpleMFRC522
reader = SimpleMFRC522()
try:
id, text = reader.read()
print(id)
print(text)
finally:
GPIO.cleanup()
```

Once you are sure you have entered the code correctly, you can save the file by pressing Ctrl + X then pressing Y and then finally hitting ENTER.
Now that we have finally finished our Read.py script we need to test it out. Before we test out the script, grab one of the RFID tags that you want to read. Once that you are ready, type the following command into your Raspberry Pi's terminal. sudo python3 Read.py With the script now running, all you need to do is place your RFID Tag on top of your RFID RC522 circuit. As soon as the Python script detects the RFID tag being placed on top, it will immediately read the data and print it back out to you. An example of what a successful output would look like is displayed below.
pi@raspberrypi:~/pi-rfid $ sudo python3 Read.py
827843705425
Pimylifeup
If you successfully receive data back from your Read.py script with the text that you pushed to the card using your Write.py script then you have successfully set up your Raspberry Pi to connect with your RFID RC522 Circuit.