

NAME: PORANDLA VINAY KUMAR VENKATESH

CLASS: TYBScIT ROLL NO: 23

ACADEMIC YEAR: 2024-2025 SEMESTER: V

**SUBJECT: INTERNET OF THINGS: THEORY AND
PRACTICE**

SUBJECT-IN-CHARGE: MS. ANSARI ARZOO

**COLLEGE: MANSI BHARAT GADA DEGREE
COLLEGE OF COMMERCE**



Oswal Shikshan & Rahat Sangh Sanchalit
**MANSI BHARAT GADA DEGREE COLLEGE OF
COMMERCE**



(Affiliated to University of Mumbai)
(NAAC Accredited 'B' Grade & ISO 9001:2015 certified)

CERTIFICATE

This is to certify that Mr. /Ms. **PORANDLA VINAY KUMAR VENKATESH** bearing **Seat. No:** _____ has satisfactorily completed the lab practical's prescribed for the **Subject: INTERNET OF THINGS: THEORY AND PRACTICE** of **Semester-V** in partial fulfillment of the requirements for the award of degree of **BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY (B.Sc. I.T.)** from University of Mumbai for the academic year: **2024-2025.**

Date:

**Signature
Internal Examiner**

**Signature
HOD (IT)**

College Seal

**Signature
External Examiner**

**Signature
I/C Principal**

INDEX

Sr.No	Date	Title	Sign
0	Practical-0		
		Starting Raspbian OS, Familiarizing with Raspberry Pi Components and interface, Connecting to ethernet, Monitor, USB	
1	Practical-1		
		Displaying different LED patterns with Raspberry Pi.	
2	Practical-2		
		Displaying Time over 4-Digit 7-Segment Display using Raspberry Pi	
3	Practical-3		
		Interfacing 16X2 LCD with Raspberry Pi to display different messages.	
4	Practical-4		
		Raspberry Pi based Oscilloscope	
5	Practical-5		
		Controlling Raspberry Pi with Telegram.	
6	Practical-6		
		Fingerprint Sensor interfacing with Raspberry Pi	
7	Practical-7		
		Raspberry Pi GPS Module Interfacing	
8	Practical-8		
		Interfacing Pi Camera with Raspberry Pi	
9	Practical-9		
		IoT based Web Controlled Home Automation using Raspberry Pi	
10	Practical-10		
		Interfacing Raspberry Pi with RFID.	

PRACTICAL 0

Aim: Starting Raspbian OS, Familiarising with Raspberry Pi Components and interface, Connecting to ethernet, Monitor, USB.

Hardware Used: Raspberry pi, Monitor, HDMI cable, Ethernet cable, USB keyboard, USB mouse, Micro-USB power supply, microSD card (32 GB), SD card reader.

Steps:

Step 1: Format the SD Card

- Download and install SD card formatter software
- Insert the microSD card into SD card reader and then insert the card reader into the computer.
- Open the SD Card Formatter software.
- Select the correct drive for the SD card.
- Click Format to clean the SD card.

Step 2: Download and Install Raspbian OS using Raspberry Pi Imager

- Download Raspberry Pi Imager: Visit the official [<https://www.raspberrypi.com/software/>]
- Click on download for windows
- Download and install Raspberry Pi Imager

Step 3: Open Raspberry Pi Imager

- Run the imager on the computer after installation.
- Click Choose device and select your Raspberry Pi model from the list (Raspberry pi 4)
- Click on Choose OS.
- Select Raspberry Pi OS (other) and then select Raspberry Pi Os full version(64-bit)
- Click on Choose Storage
- Select the formatted SD card as the target for installation.

Step 4: OS Customization Settings

- In a popup, Imager will ask us to apply OS customisation. Click the Edit Settings button to open OS customisation.
- Set a custom hostname for the Raspberry Pi (optional).
- Set username and password
- Enable SSH and set a password for the default user.
- Set up Wi-Fi by entering the SSID and password of your network (optional for Ethernet users).
- Choose your language, country, and time zone.
- Click Save after customizing.

Step 5: Write the OS:

- After selecting the OS, storage, and customizations, click Write to begin installing the OS onto the SD card.
- Once complete, safely eject the SD card.

Step 6: Boot the Raspberry Pi

- Insert the SD card into the Raspberry Pi.
- Connect a Monitor using an HDMI cable.
- Connect a USB keyboard and USB mouse to the Raspberry Pi.
- Connect the Ethernet cable (optional, for wired internet).
- Plug in the Power Supply (USB-C or micro-USB) to boot the Raspberry Pi.

The Raspberry Pi will boot up and display the Raspbian OS welcome screen on the monitor.

PRACTICAL 01

Aim: Displaying different LED patterns with Raspberry Pi.

Hardware Used: Raspberry Pi, LED Module with Resistors, Connecting Wires.

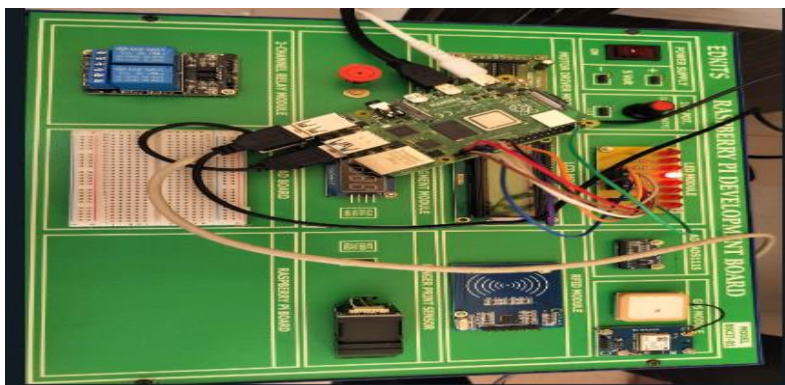
Connection:

LED Module	Raspberry Pi
Ground	06
led1	29
led2	31
led3	33
led4	35
led5	36
led6	37
led7	38
led8	40

Steps:

- Connect the raspberry pi to the PC
- Connect the ground pin of the LED Module to pin no. 6 of the Raspberry Pi (ground pin).
- Connect each LED in the module to its designated GPIO pins on the Raspberry Pi according to the specified configuration
- Turn on the Raspberry Pi by providing power through the power supply
- In the top-left corner, click on the Raspberry Pi icon.
- Navigate to Programming and select Thonny Python IDE.
- Create a new file in Thonny.
- Write your code in the editor.
- Save the file with a .py extension.
- Run the file to execute the Python code.

Connection Diagram:



Code:**Filename: LedPattern.py**

```
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
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)

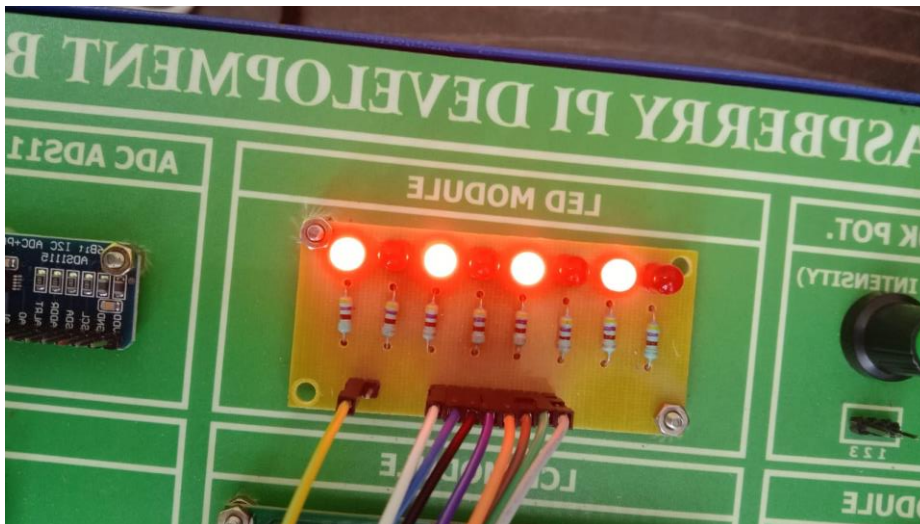
```

```

try:
    while True:
        patternOne()
        patternTwo()
        patternThree()
        patternFour()
        patternFive()
finally:
    GPIO.cleanup()

```

Output:



PRACTICAL 02

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

Hardware Used: Raspberry Pi, Seven Segment Module , Connecting Wires.

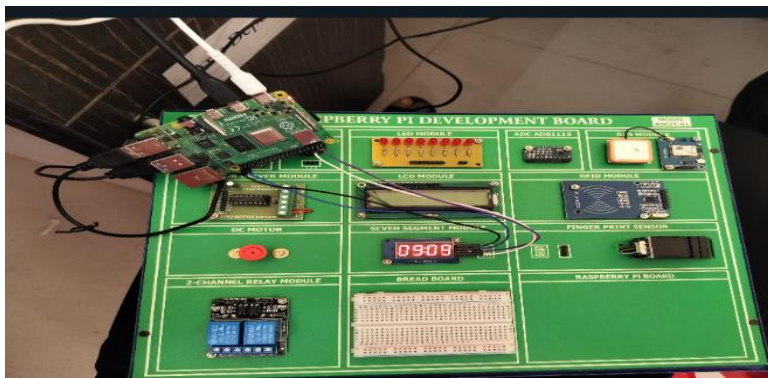
Connection:

Seven Segment Module	Raspberry Pi
Clk	38 40
DIO	40 38
Vcc	2
Ground	6

Steps:

- To download libraries, open Web Browser on Raspberry Pi and log on to the following link: <https://github.com/timwaizenegger/raspberrypi-examples/tree/master/actor-led-7segment-4numbers>.
- Click on actor-led-7segment-4numbers.zip folder and then click on download button to download the file.
- Now, on Raspberry Pi move to the location to find the zip file downloaded and unzip the file.
- Now, open Thonny Python IDE in Programming option which is present in the Raspberry icon.
- Write the code given below and save it in the same folder i.e. actor-led-7segment-4numbers. Since, the code is dependent on tm1637.py file which is present in the same folder.
- Run the file to execute the code.

Connection Diagram:



Code:

Filename: SevenSegment.py

```
from time import sleep
import RPi.GPIO as GPIO
import tm1637

try:
    import thread
except ImportError:
    import _thread as thread

# Initialize the clock (GND, VCC-3.3V, Example Pins are DIO-20 and CLK-21)
Display = tm1637.TM1637(CLK=21, DIO=20, brightness=1.0)

try:
    print("Starting clock in the background (press CTRL + C to stop)")

    Display.StartClock(military_time=True)
    Display.SetBrightness(1.0)

    while True:
        Display.ShowDoublepoint(True)
        sleep(1)
        Display.ShowDoublepoint(False)
        sleep(1)

except KeyboardInterrupt:
    print("Properly closing the clock and open GPIO pins")
    Display.StopClock()
    Display.cleanup()
```

Output:



PRACTICAL 03

Aim: Interfacing 16*2 LCD with Raspberry Pi to display different messages.

Hardware Used: : Raspberry Pi, LCD Module , Connecting Wires.

Connection:

16*2 LCD Module	Raspberry Pi
1	6(Ground)
2	2(Power)
4	37
5	9(Ground)
6	35
11	22
12	18
13	15
14	13
15	17
16	20

10K Potentiometer	Raspberry Pi/LCD Module
1	4(Raspberry Pi(power))
2	3 (LCD pin)
3	39(Raspberry Pi (ground))

Steps:

Write commands in terminal on Raspberry Pi:

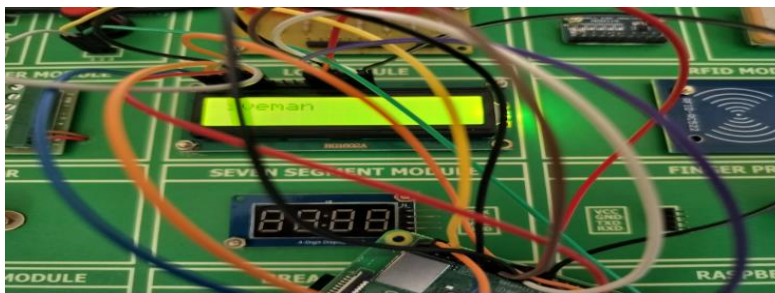
First, activate the virtual environment created earlier

- source myenv/bin/activate
- python -m venv --system-site-packages venv

Modules to install

- sudo apt install python3-pip
 - pip3 install adafruit-blinka
 - pip3 install adafruit-circuitpython-charlcd.
- Install the libraries to provide support for LCD module in Python to display messages.
 - Create a new python file “lcd.py” at myenv\lib\python 3.11\site- packages and copy the following code in it.
 - Run the file to execute the code.

Connection Diagram:



Code:

Filename: LCDDisplay.py

```
import time
import board
import digitalio
import adafruit_character_lcd.character_lcd as characterlcd
# Define LCD column and row size for 16x2 LCD.
lcd_columns = 16
lcd_lines = 2
# Raspberry Pi pin configuration:
lcd_rs = digitalio.DigitalInOut(board.D26)
lcd_en = digitalio.DigitalInOut(board.D19)
lcd_d4 = digitalio.DigitalInOut(board.D25)
lcd_d5 = digitalio.DigitalInOut(board.D24)
lcd_d6 = digitalio.DigitalInOut(board.D22)
lcd_d7 = digitalio.DigitalInOut(board.D27)
lcd_backlight = digitalio.DigitalInOut(board.D4)
# Initialize the LCD Class
lcd = characterlcd.Character_LCD_Mono(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7,
lcd_columns, lcd_lines, lcd_backlight)
# Print a two-line message
lcd.message = "Edkits\nElectronics"
time.sleep(5)
# Clear the display
lcd.clear()
```

Output:



PRACTICAL 04

Aim: Raspberry Pi based Oscilloscope.

Hardware Used: Raspberry Pi, ADC Module , Connecting Wires.

Connection:

ADC Module	Raspberry Pi
Vcc	1(3.3V)
Ground	6
SCL	5
SDA	3

Steps:

Write commands in terminal on Raspberry Pi:

First, activate the virtual environment created earlier

- source myenv/bin/activate
- python -m venv --system-site-packages venv

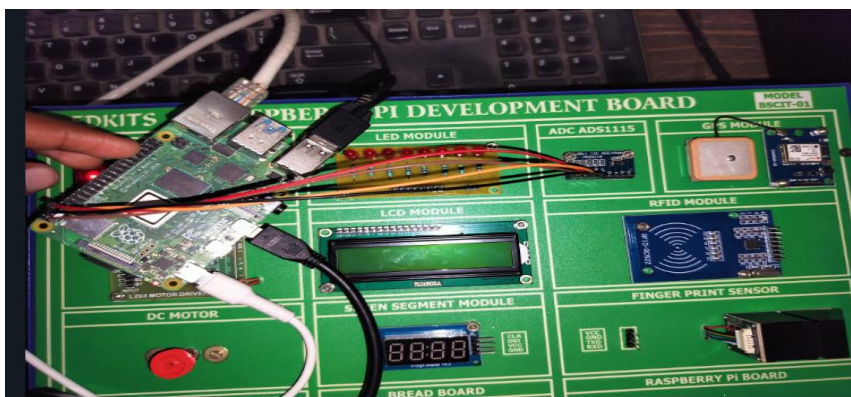
Modules to install

- sudo apt-get update
- sudo apt-get upgrade
- sudo apt-get install build-essential python3-dev python3-smbus git python3-pip
- pip3 install Adafruit-ADS1x15 matplotlib drawnow
- pip3 install smbus2 adafruit-ads1x15
- pip3 install adafruit-circuitpython-ads1x15
- sudo apt-get install -y i2c-tools

Steps:

- To enable I2C interface communication, run the command i.e. sudo raspi-config in terminal on Raspberry Pi.
- When the configuration panels open, select the Interfacing Options and then choose I2C option.
- Select Yes to enable I2C communication.
- Click on Finish.

Connection Diagram:



Code:

Filename: Oscilloscope.py

```
import time
import matplotlib.pyplot as plt
from drawnow import drawnow
from adafruit_ads1x15.analog_in import AnalogIn
import adafruit_ads1x15.ads1115 as ADS
from smbus2 import SMBus
import board
import busio

# Initialize I2C bus manually
i2c = busio.I2C(board.SCL, board.SDA)

# Create an ADS1115 instance
ads = ADS.ADS1115(i2c)

# Gain setting for the ADS1115
GAIN = 1
val = []
cnt = 0
max_points = 50 # Maximum number of points to display in the plot

plt.ion() # Enable interactive mode for Matplotlib

# Create the figure function for drawing the plot
def makeFig():
    plt.clf() # Clear the current figure to avoid overlap
    plt.ylim(-5000, 5000) # Set y-axis limits according to expected values
    plt.title('Oscilloscope - ADC Readings')
    plt.grid(True)
    plt.ylabel('ADC Outputs')
    plt.xlabel('Sample Number')
    plt.plot(val, 'ro-', label='Channel 0') # Plot the values in red with markers
    plt.legend(loc='lower right')

# Start continuous ADC readings on channel 0
print('Reading ADS1115 channel 0')

try:
    while True:
        # Read the ADC value from channel 0
        chan = AnalogIn(ads, ADS.P0) # Reading from channel 0 with the specified gain
        value = chan.value
        print(f'Channel 0: {value}')

        # Append the ADC value to the list for plotting
        val.append(int(value))

        # Update the plot with the new data
        drawnow(makeFig)
        plt.pause(0.1) # Pause for a short moment to update the plot
```

```

# Maintain a fixed number of values for display
if len(val) > max_points:
    val.pop(0) # Remove the oldest value from the list to keep it within the limit

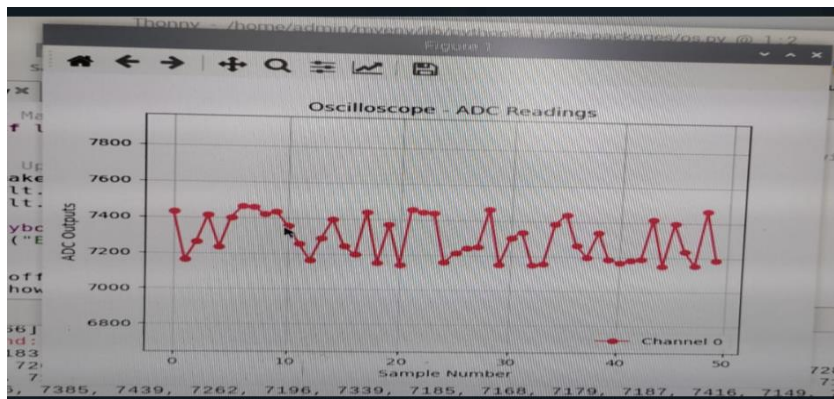
# Sleep for a short period before the next reading (adjust if necessary)
time.sleep(0.5)

except KeyboardInterrupt:
    print("Exiting the program...")

finally:
    plt.ioff() # Disable interactive mode to finalize the plot
    plt.show() # Show the final plot after exiting the loop

```

Output:



PRACTICAL 05

Aim: Controlling Raspberry Pi with Telegram.

Hardware Used: Raspberry Pi, LED Module with Resistors, Connecting Wires.

Connection:

LED Module	Raspberry Pi
1	40
GND(ground)	6

Steps:

Write commands in terminal on Raspberry Pi:

Create the virtual environment

- `python -m venv myenv`
- `source myenv/bin/activate`
- `pip list`
- `python -m venv --system-site-packages venv`

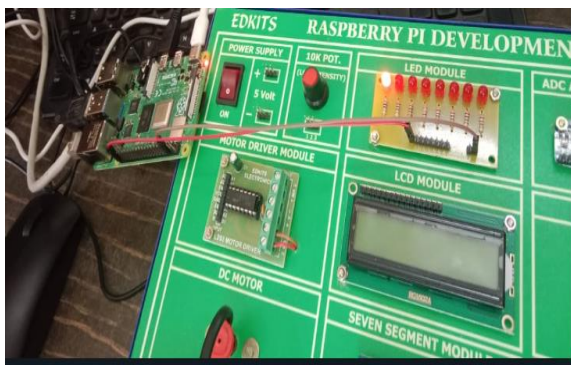
Modules to install

- `sudo apt-get install python3-pip`
- `pip3 install telepot`

Steps:

- Install Telegram App on mobile phone.
- Open Telegram on phone, search for BotFather and select it.
- Type `/start` to chat with the bot in BotFather.
- To obtain a bot account, text `/newbot` in BotFather.
- It will ask to choose a name for the bot.
- Then, it will ask to choose a username for the bot.
- At the end of the process, it will give a token which represents the bot account. This token will be used in code on Raspberry Pi.
- Create a new python file "Telegram.py" at `myenv\lib\python 3.11\site-packages` and copy the following code in it.
- Run the file to execute the code.

Connection Diagram:



Code:

Filename:Telepot.py

```
import sys
import time
import random
import datetime
import telepot
import RPi.GPIO as GPIO
from telepot.loop import MessageLoop

red = 40 # connect red LED at pin 40

now = datetime.datetime.now()

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
GPIO.setup(red, GPIO.OUT)
GPIO.output(red, 0)

def action(msg):
    chat_id = msg['chat']['id']
    command = msg['text']
    print('Got command: %s' % command)

    if 'On' in command:
        message = "Turn On"
        message = message + " red"
        GPIO.output(red, 1)
        bot.sendMessage(chat_id, message)

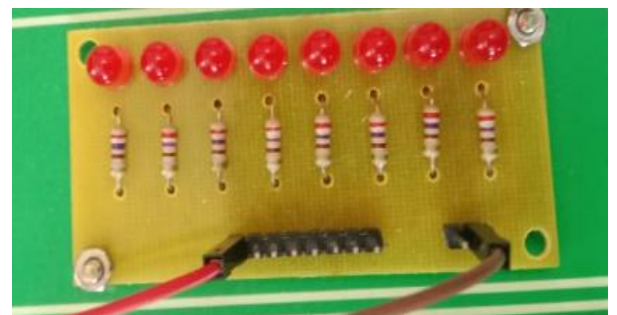
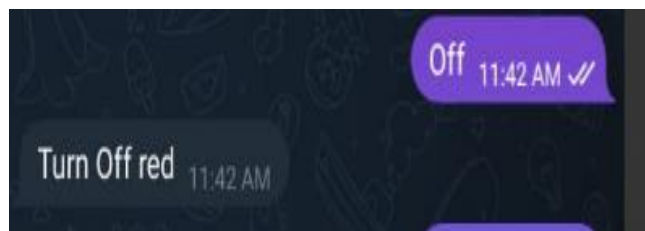
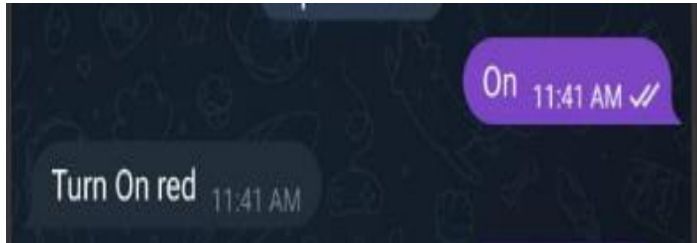
    if 'Off' in command:
        message = "Turn Off"
        message = message + " red"
        GPIO.output(red, 0)
        bot.sendMessage(chat_id, message)

bot = telepot.Bot('626665131:AAHsNzQbqSj9GZ9-w2t41')
print(bot.getMe())

MessageLoop(bot, action).run_as_thread()
print('I am listening...')

while 1:
    time.sleep(10)
```

Output:



PRACTICAL 06

Aim: Fingerprint Sensor interfacing with Raspberry Pi.

Hardware Used: Raspberry Pi, Finger Print Sensor Module , Connecting Wires.

Connection:

Finger Print Sensor Module	Raspberry Pi
V _{cc}	2(Power)
GND(ground)	6
T _{XD}	8(Serial Communication)
R _{XD}	10(Serial Communication)

Steps:

Write commands in terminal on Raspberry Pi:

First, activate the virtual environment created earlier

- `source myenv/bin/activate`
- `python -m venv --system-site-packages venv`

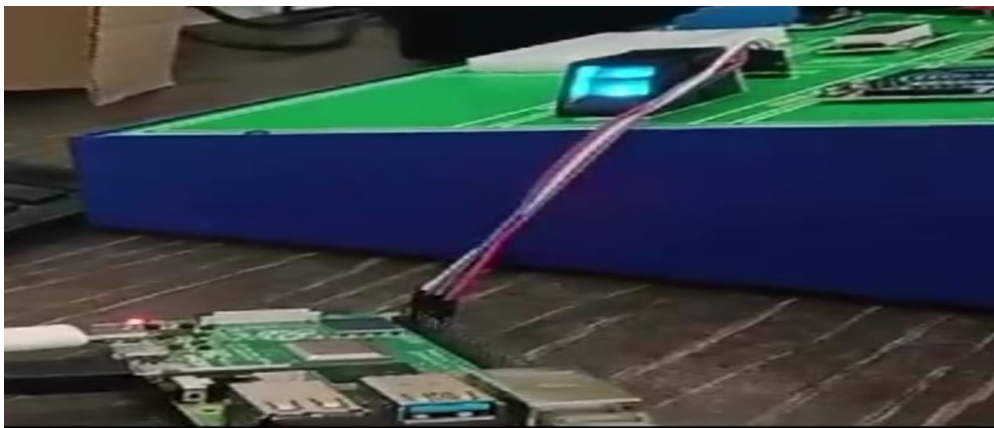
Modules to install

- `pip3 install adafruit-circuitpython-fingerprint`

Steps:

- To enable serial communication, run the command i.e. `sudo raspi-config` in terminal on Raspberry Pi.
- Select the Interfacing Options and then choose Serial option.
- Select No on enabling the login shell.
- Select Yes on enabling serial port hardware.
- Once completed monitor will display like this:
- The serial login shell is disabled.
- The serial interface is enabled.
- Then select the reboot option.

Connection Diagram:



Code:

Filename:Fingerprint.py

```
import time
import board
from digitalio import DigitalInOut, Direction
import adafruit_fingerprint

# Set up LED
led = DigitalInOut(board.D13)
led.direction = Direction.OUTPUT

# If using with Linux/Raspberry Pi and hardware UART:
import serial
uart = serial.Serial("/dev/ttyS0", baudrate=57600, timeout=1)
finger = adafruit_fingerprint.Adafruit_Fingerprint(uart)

def get_fingerprint():
    """Get a finger print image, template it, and see if it matches!"""
    print("Waiting for image...")
    while finger.get_image() != adafruit_fingerprint.OK:
        pass

    print("Templating...")
    if finger.image_2_tz(1) != adafruit_fingerprint.OK:
        return False

    print("Searching...")
    if finger.finger_search() != adafruit_fingerprint.OK:
        return False

    return True

# pylint: disable-too-many-branches
def get_fingerprint_detail():
    """Get a finger print image, template it, and see if it matches!
    This time, print out each error instead of just returning on failure"""

    print("Getting image...", end="")
    i = finger.get_image()

    if i == adafruit_fingerprint.OK:
        print("Image taken")
    else:
        if i == adafruit_fingerprint.NOFINGER:
            print("No finger detected")
        elif i == adafruit_fingerprint.IMAGEFAIL:
            print("Imaging error")
        else:
            print("Other error")
        return False

    print("Templating...", end="")
```

```

i = finger.image_2_tz(1)

if i == adafruit_fingerprint.OK:
    print("Templated")
else:
    if i == adafruit_fingerprint.IMAGEMESS:
        print("Image too messy")
    elif i == adafruit_fingerprint.FEATUREFAIL:
        print("Could not identify features")
    elif i == adafruit_fingerprint.INVALIDIMAGE:
        print("Image invalid")
    else:
        print("Other error")
    return False

print("Searching...", end="")
i = finger.finger_fast_search()

if i == adafruit_fingerprint.OK:
    print("Found fingerprint!")
    return True
elif i == adafruit_fingerprint.NOTFOUND:
    print("No match found")
else:
    print("Other error")
    return False

# pylint: disable-too-many-statements
def enroll_finger(location):
    """Take 2 finger images and template them, then store in 'location'"""

    for fingerimg in range(1, 3):
        if fingerimg == 1:
            print("Place finger on sensor...", end="")
        else:
            print("Place same finger again...", end="")

        while True:
            i = finger.get_image()
            if i == adafruit_fingerprint.OK:
                print("Image taken")
                break
            elif i == adafruit_fingerprint.NOFINGER:
                print(".", end="")
            elif i == adafruit_fingerprint.IMAGEFAIL:
                print("Imaging error")
                return False
            else:
                print("Other error")
                return False

        print("Templating...", end="")
        i = finger.image_2_tz(fingerimg)

```

```

if i == adafruit_fingerprint.OK:
    print("Templated")
else:
    if i == adafruit_fingerprint.IMAGEMESS:
        print("Image too messy")
    elif i == adafruit_fingerprint.FEATUREFAIL:
        print("Could not identify features")
    elif i == adafruit_fingerprint.INVALIDIMAGE:
        print("Image invalid")
    else:
        print("Other error")
    return False

if fingerimg == 1:
    print("Remove finger")
    time.sleep(1)
    while finger.get_image() != adafruit_fingerprint.NOFINGER:
        pass

print("Creating model...", end="")
i = finger.create_model()
if i == adafruit_fingerprint.OK:
    print("Created")
elif i == adafruit_fingerprint.ENROLLMISMATCH:
    print("Prints did not match")
    return False
else:
    print("Other error")
    return False

print("Storing model #%%d..." % location, end="")
i = finger.store_model(location)
if i == adafruit_fingerprint.OK:
    print("Stored")
elif i == adafruit_fingerprint.BADLOCATION:
    print("Bad storage location")
    return False
elif i == adafruit_fingerprint.FLASHERR:
    print("Flash storage error")
    return False
else:
    print("Other error")
    return False

return True

def get_num():
    """Use input() to get a valid number from 1 to 127. Retry till success!"""
    i = 0
    while (i > 127) or (i < 1):
        try:
            i = int(input("Enter ID # from 1-127: "))
        except ValueError:
            pass

```

```
return i
```

```
while True:
```

```
    print("\nFingerprint templates:", finger.templates)
```

```
    print("e) enroll print")
```

```
    print("d) delete print")
```

```
    print("f) find print")
```

```
    print("q) quit")
```

```
    c = input("> ")
```

```
    if c == "e":
```

```
        enroll_finger(get_num())
```

```
    elif c == "f":
```

```
        if get_fingerprint():
```

```
            print("Detected #", finger.finger_id, "with confidence", finger.confidence)
```

```
        else:
```

```
            print("Finger not found")
```

```
    elif c == "d":
```

```
        if finger.delete_model(get_num()) == adafruit_fingerprint.OK:
```

```
            print("Deleted!")
```

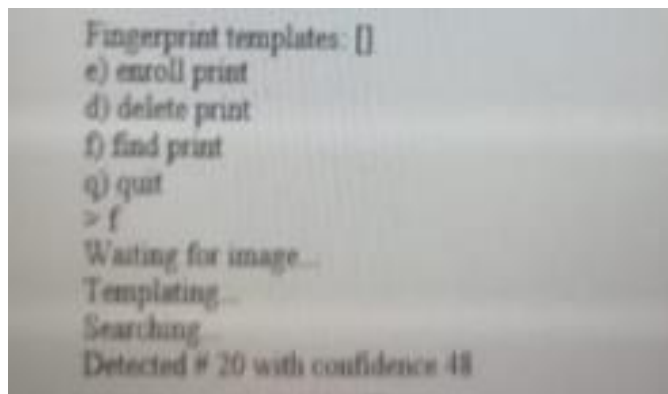
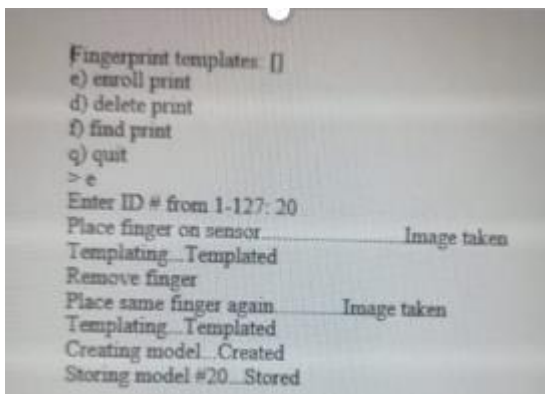
```
        else:
```

```
            print("Failed to delete")
```

```
    elif c == "q":
```

```
        break
```

Output:



PRACTICAL 07

Aim: Raspberry Pi GPS Module Interfacing

Hardware Requirements: Raspberry Pi, SD Card, Ethernet Cable / Wi-Fi, Power Supply to Pi, Neo 6m v2 GPS Module, Jumper wires, Breadboard, 2x16 LCD Display, Potentiometer.

Connection:

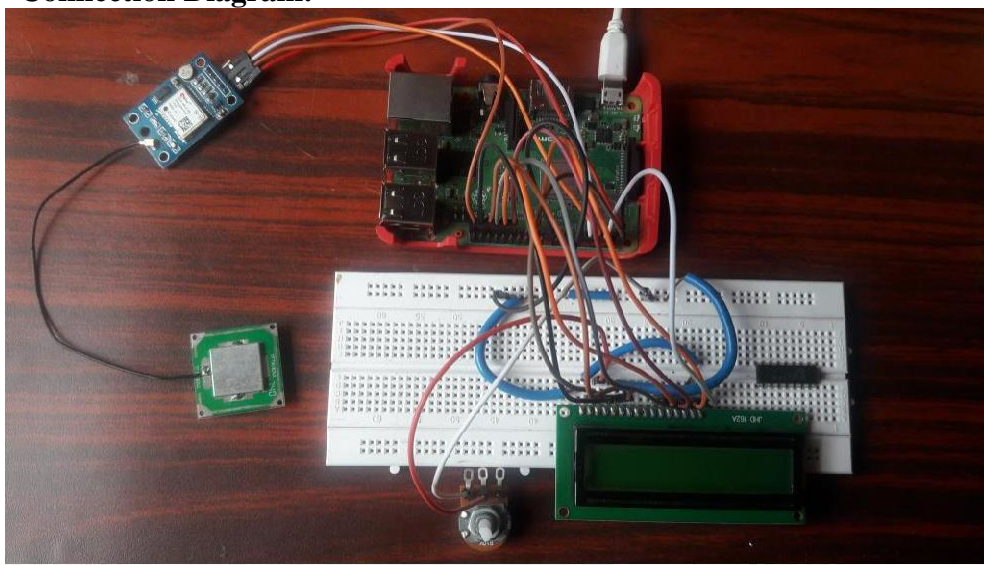
Connect GPS Module to RPi:

Neo 6m V2 GPSBoard Pin	Details	Raspberry Pi Physical Pin
VCC	Power	1
GND	Common Ground	39
TXD	Data Output	10
RXD	Data Input	8

Connect 2x16 LCD Display to RPi:


Sr No	LCD Display Board Pin	RPI Physical Pin
4	RS	37
6	E	35
1 1	D4	33
1 2	D5	31
1 3	D6	29
1 4	D7	23

Connection Diagram:




Steps:

Step 1: Update Raspberry Pi



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo apt-get update  
  
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo apt-get upgrade
```

Step 2: edit the /boot/config.txt file



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo nano /boot/config.txt
```

At the bottom of this file, add the above lines

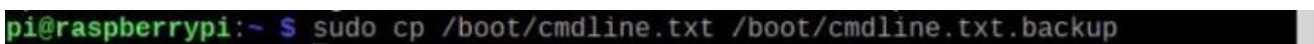


```
dtoverlay=pi3-disable-bt  
core_freq=250  
enable_uart=1  
force_turbo=1
```

The `dtoverlay=pi3-disable-bt` disconnects the bluetooth from the `ttyAMA0`, this is to allow us access to use the full UART power available via `ttyAMA0` instead of the mini UART `ttyS0`.

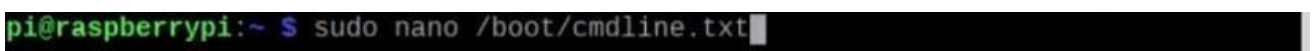
Save with Ctrl+X, yes and press Enter

Step 3: Before proceeding, make a copy of cmdline.txt file



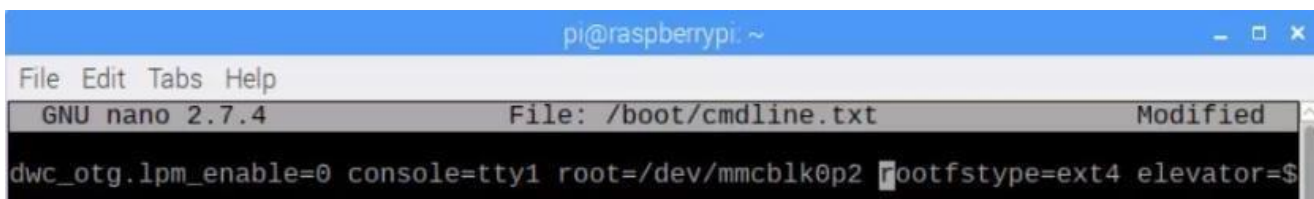
```
pi@raspberrypi:~ $ sudo cp /boot/cmdline.txt /boot/cmdline.txt.backup
```

Now, edit file cmdline.txt



```
pi@raspberrypi:~ $ sudo nano /boot/cmdline.txt
```

Remove `console=serial0,115200` and Modify `root=/dev/mmcblk0p2`



```
pi@raspberrypi: ~  
File Edit Tabs Help  
GNU nano 2.7.4 File: /boot/cmdline.txt Modified  
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=...
```

Save with Ctrl+X, yes and press Enter.

Step 4: Reboot Raspberry Pi using the command `sudo reboot`

Step 5: Stop and disable the Pi's serial ttyS0 service

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~$ sudo systemctl stop serial-getty@ttyS0.service  
pi@raspberrypi:~$ sudo systemctl disable serial-getty@ttyS0.service
```

The following commands can be used to enable it again if needed

```
sudo systemctl enable serial-getty@ttyS0.service
```

```
sudo systemctl start serial-getty@ttyS0.service
```

Step 6: Reboot Raspberry Pi using the command *sudo reboot*

Step 7: Now, Enable the ttyAMA0 service

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~$ sudo systemctl enable serial-getty@ttyAMA0.service
```

Verify it using *ls -l /dev* command

```
lrwxrwxrwx 1 root root      7 Sep 20 12:39 serial0 -> ttyAMA0  
lrwxrwxrwx 1 root root      5 Sep 20 12:39 serial1 -> ttyS0
```

Step 8: Install minicom and pynmea2

Install minicom package which is used to connect to the GPS module and make sense of the data.

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~$ sudo apt-get install minicom
```

Install pynmea2 library which is used to parse the received data.

```
pi@raspberrypi:~$ sudo pip install pynmea2
```

Step 9: Use minicom command to test our GPS module is working fine.

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~$ sudo minicom -D /dev/ttyAMA0 -b 9600
```

9600 represents the baud rate at which the GPS module communicates.

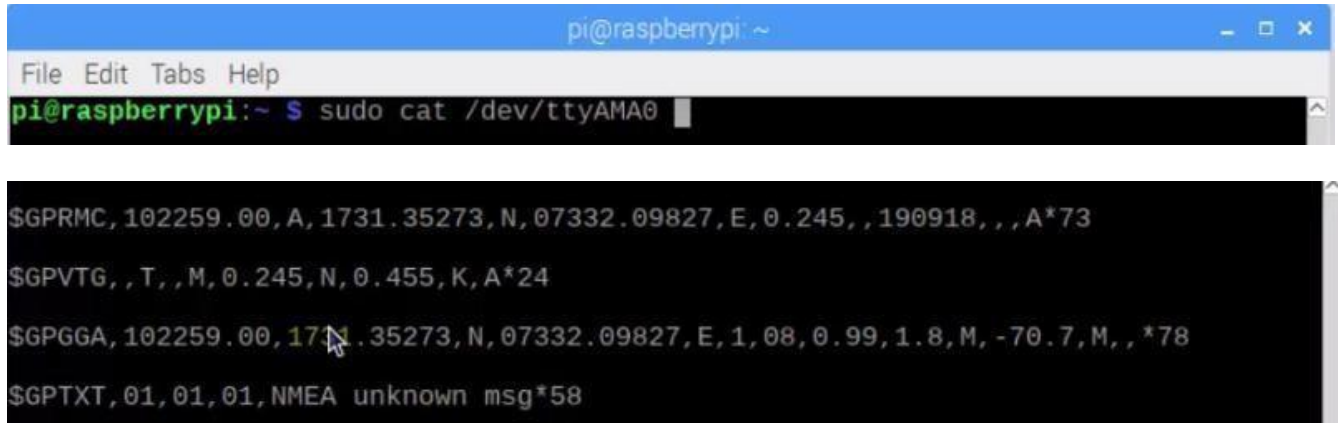
Here, we can see NMEA sentences .NMEA format consist several sentences, in which we only need one sentence. This sentence starts from \$GPGGA and contains the coordinates

```
$GPTXT,01,01,01,NMEA unknown msg*58  
$GPTXT,01,01,01,NMEA unknown msg*58  
$GPRMC,101752.00,A,1731.35655,N,07332.09973,E,0.503,,190918,,A*7B  
$GPVTG,,T,,M,0.503,N,0.932,K,A*2D  
$GPGGA,101752.00,1731.35655,N,07332.09973,E,1,07,1.04,2.0,M,-70.7,M,,*74  
$GPGSA,A,3,2120,10,2
```

Sometimes we received sentences which contains unknown msg*58, but this is not an error, actually it may takes some time to track your GPS module. (Even for the first time more than 20-30 minutes.)

I suggest keep your GPS module's antenna in open space (e.g. near the window) To exit from above window, Press Ctrl+A, and Press x and Enter Key

Step 10: The above same test can also be done using cat command



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo cat /dev/ttyAMA0  
$GPRMC,102259.00,A,1731.35273,N,07332.09827,E,0.245,,190918,,A*73  
$GPVTG,,T,,M,0.245,N,0.455,K,A*24  
$GPGGA,102259.00,1731.35273,N,07332.09827,E,1,08,0.99,1.8,M,-70.7,M,,*78  
$GPTXT,01,01,01,NMEA unknown msg*58
```

This sentence gives you Latitude found after two commas and Longitude found after four commas

Step 11: Write Python script to display your current Latitude and Longitude on LCD Display

Code:

```
import time import serial import string import pynmea2  
import RPi.GPIO as gpio  
import Adafruit_CharLCD as LCD gpio.setmode(gpio.BCM)  
lcd = LCD.Adafruit_CharLCD(rs=26, en=19, d4=13, d5=6, d6=5, d7=11,  
cols=16, lines=2)  
lcd.message("MSD Gurukul\n Welcomes You") time.sleep(2)  
lcd.clear() lcd.message("GPS Demo") time.sleep(2)  
lcd.clear()  
port = "/dev/ttyAMA0" # the serial port to which the pi is connected. #create a serial object  
ser = serial.Serial(port, baudrate = 9600, timeout = 0.5) try:  
while 1:  
try:  
data = ser.readline() except:  
print("loading")  
#wait for the serial port to churn out data  
if data[0:6] == '$GPGGA': # the long and lat data are always contained in the GPGGA string of the  
NMEA data  
  
msg = pynmea2.parse(data)  
latval = msg.lat #parse the latitude and print concatlat = "Lat:" + str(latval) print(concatlat)  
lcd.set_cursor(0,0) lcd.message(concatlat) #parse the longitude and print longval = msg.lon  
concatlong = "Long:" + str(longval) print(concatlong) lcd.set_cursor(0,1) lcd.message(concatlong)  
time.sleep(0.5)#wait a little before picking the next data. except KeyboardInterrupt:  
lcd.clear() lcd.message("Thank You") time.sleep(2)
```

Output:

```
$GPRMC,175839.00,V,,,,,200820,,,N*74
$GPVTG,,,,,,N*30
$GPGGA,175839.00,,,,,0,03,5.51,,,,,*55
$GPGSA,A,1,22,16,04,,,,,,5.60,5.51,1.00*00
$GPGSV,2,1,07,01,03.187,,03,,26,04,01,015,29,09,,23*72
$GPGSV,2,2,07,16,45.036,21,22,32,100,31,39,32,253,34*4C
$GPGLL,,,,,175839.00,V,N*4B
$GPRMC,175840.00,V,,,,,200820,,,N*7A
$GPVTG,,,,,,N*30
$GPGGA,175840.00,,,,,0,03,5.51,,,,,*58
$GPGSA,A,1,22,16,04,,,,,,5.60,5.51,1.00*00
```

PRACTICAL 08

Aim: Interfacing Pi Camera with Raspberry Pi.

Hardware Used: Raspberry Pi, Camera Module , Connecting Wires.

Connection:

Connect the Camera module carefully to the Camera port of the Raspberry Pi Board. Make sure the Raspberry Pi is Off at the time of connection.

Steps:

Write commands in terminal on Raspberry Pi:

First, activate the virtual environment created earlier

- source myenv/bin/activate
- python -m venv --system-site-packages venv

Modules to install

- sudo apt update
- sudo apt install -y python3-picamera2
- sudo apt install -y libcamera-apps python3-libcamera

Steps:

- Install the libraries to support the Pi Camera.
- Create a new python file “picamera.py” at myenv\lib\python 3.11\site- packages and copy the following code in it.
- Run the file to execute the code.
- It will capture the image and store in jpg format located at myenv\lib\python 3.11\site- packages where the python file is created.

Connection Diagram:



Code:

Filename: CameraPractical.py

```
from picamera2 import Picamera2, Preview
import time
```

```
picam2 = Picamera2()
picam2.configure(picam2.create_preview_configuration(main={"size": (640, 480)}))
picam2.start_preview(Preview.QTGL)
picam2.start()
time.sleep(5)
picam2.capture_file("image.jpg")
picam2.stop_preview()
picam2.stop()
print("Image captured.")
```

Output:

PRACTICAL 09

Aim: IoT based Web Controlled Home Automation using Raspberry Pi.

Hardware Used: Raspberry Pi, LED Module with Resistors, Bulb, Connecting Wires.

Connection:

2-Channel Relay Module	Raspberry Pi
Vcc	2
Ground	6
IN1(SoC Pin 27)	13

LED Module	Raspberry Pi
Vcc	11
Ground	9

Steps:

Write commands in terminal on Raspberry Pi:

First, activate the virtual environment created earlier

- source myenv/bin/activate
- python -m venv --system-site-packages venv

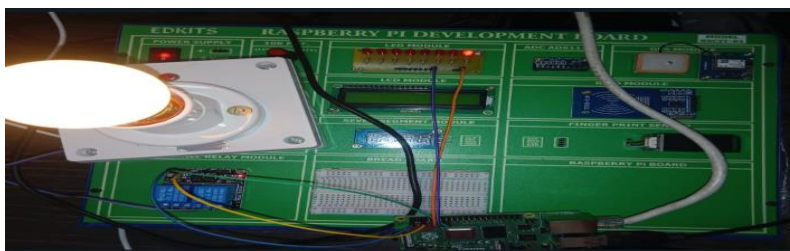
Modules to install

- sudo apt update
- sudo apt install python3-flask

Steps:

- Create a new html file “index.html” in templates folder located at myenv\lib\python 3.11\site- packages and copy the HTML code in it.
- Create a new python file “homeauto.py” at myenv\lib\python 3.11\site- packages and copy the following code in it.
- Run the file to execute the code.
- When code is executed, it will display 2 IP addresses in the monitor screen.
- One is used in web to display the interface created in html file.
- Other is used in Ethernet, as Ethernet cable is connected in Ethernet port on Raspberry Pi.
- Copy the first IP address and paste it in the Web Browser.

Connection Diagram:



Code:**Filename: index.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>LED Control</title>
</head>
<body>
  <h1>Web Controlled LED</h1>
  <form action="/led/on" method="get">
    <button type="submit">Turn LED ON</button>
  </form>
  <form action="/led/off" method="get">
    <button type="submit">Turn LED OFF</button>
  </form>
</body>
</html>
```

Filename: HomeAutomation.py

```
from flask import Flask, render_template, request
import RPi.GPIO as GPIO

# GPIO Setup
LED_PIN = 17 # Pin to control LED (previous setup)
RELAY_PIN = 27 # Pin to control relay (connected to IN1 of the relay module)
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)
GPIO.setup(RELAY_PIN, GPIO.OUT)

# Create Flask app
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/led/<state>')
def led_control(state):
    if state == "on":
        GPIO.output(LED_PIN, GPIO.HIGH)
        return "LED is ON"
    elif state == "off":
        GPIO.output(LED_PIN, GPIO.LOW)
        return "LED is OFF"
    else:
        return "Invalid command"

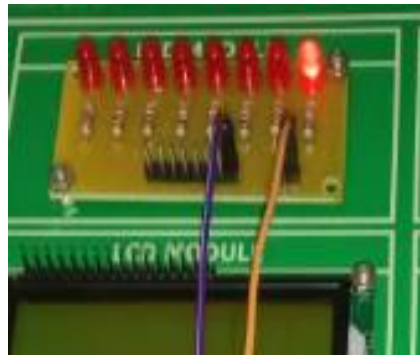
@app.route('/relay/<state>')
def relay_control(state):
    if state == "on":
```



```
GPIO.output(RELAY_PIN, GPIO.LOW) # Activates relay (turns bulb ON)
return "Bulb is ON"
elif state == "off":
    GPIO.output(RELAY_PIN, GPIO.HIGH) # Deactivates relay (turns bulb OFF)
    return "Bulb is OFF"
else:
    return "Invalid command"

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000)
```

Output:



PRACTICAL 10

Aim: Interfacing Raspberry Pi with RFID.

Hardware Used: Raspberry Pi, RFID Reader (RC522), RFID Tags or Cards, Breadboard, Jumper wires (Female to Male)

Connection

RFID Reader Board Pin	RPI Physical Pin	Raspberry Function
SDA	24	GPIO8 (SPI_CE0_N)
SCK	23	GPIO11 (SPI0_CLK)
MOSI	19	GPIO10 (SPI0_MOSI)
MISO	21	GPIO9 (SPI0_MISO)
IRQ	UNUSED	
GND	6	GND
RST	22	GPIO25 (GPIO_GEN6)
3.3V	1	3.3V PWR

Connection Diagram:



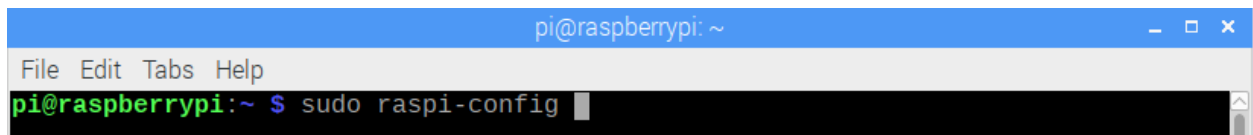
Steps:

Step 1: Update Raspberry Pi

Step 2: Enable SPI Interface using `sudo raspi-config` tool

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo raspi-config
```

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo apt-get upgrade
```

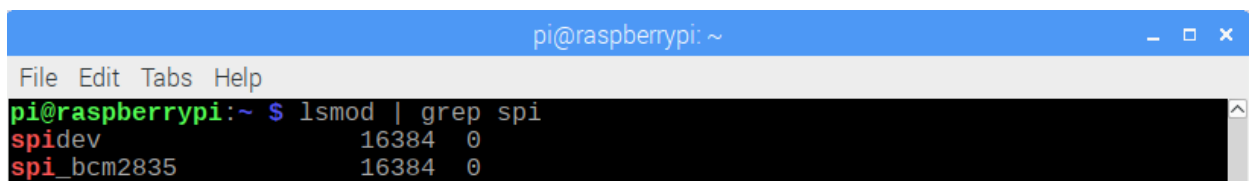


```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo raspi-config
```

Select using Arrow Keys ☐ Interfacing Options ☐ SPI ☐ Enable

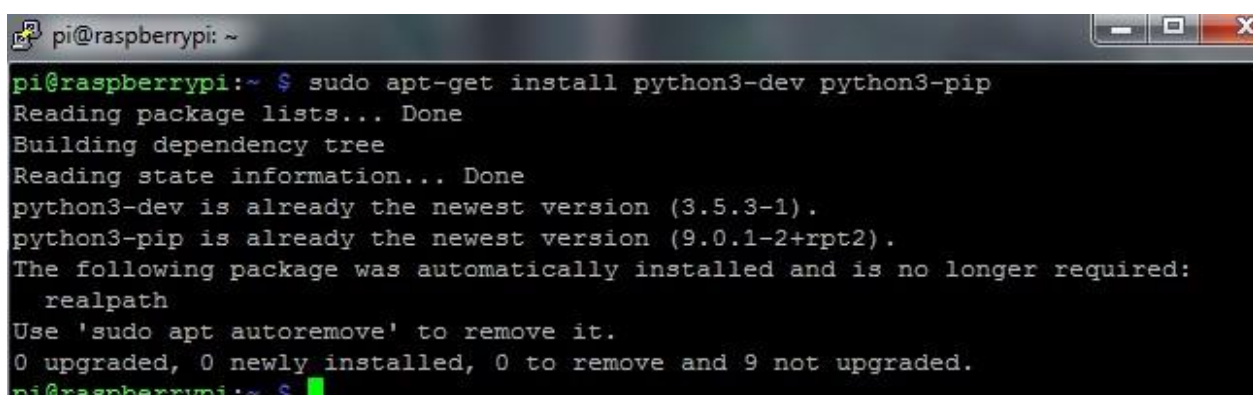
Step 3: Reboot Raspberry Pi

Step 4: Check to make sure that SPI has been enabled.



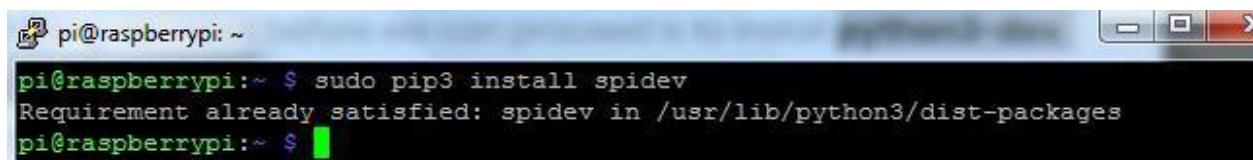
```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ lsmod | grep spi  
spidev                16384  0  
spi_bcm2835           16384  0
```

Step 5: Install python3-dev, python3-pip packages to setting up RFID reader



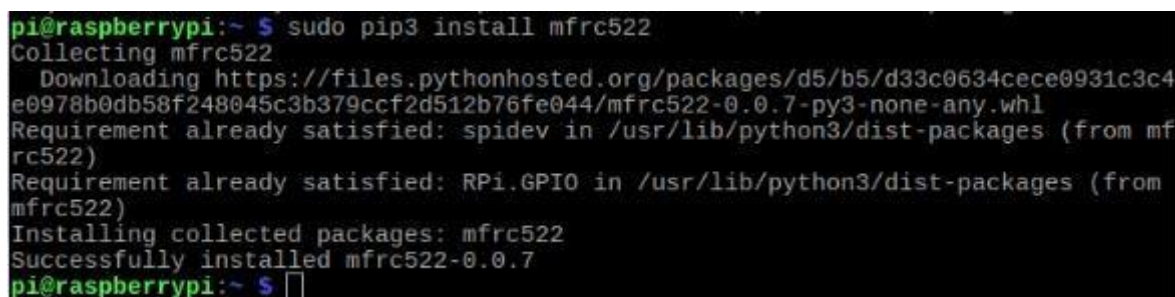
```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ sudo apt-get install python3-dev python3-pip  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
python3-dev is already the newest version (3.5.3-1).  
python3-pip is already the newest version (9.0.1-2+rpt2).  
The following package was automatically installed and is no longer required:  
  realpath  
Use 'sudo apt autoremove' to remove it.  
0 upgraded, 0 newly installed, 0 to remove and 9 not upgraded.  
pi@raspberrypi:~ $
```

Step 6: Install spidev to Raspberry Pi using pip. The spidev library helps to handle interactions with the SPI



```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ sudo pip3 install spidev  
Requirement already satisfied: spidev in /usr/lib/python3/dist-packages  
pi@raspberrypi:~ $
```

Step 7: Install the MFRC522 library using pip that helps talk to the RC522 module over the SPI interface



```
pi@raspberrypi:~ $ sudo pip3 install mfrc522  
Collecting mfrc522  
  Downloading https://files.pythonhosted.org/packages/d5/b5/d33c0634cece0931c3c4e0978b0db58f248045c3b379ccf2d512b76fe044/mfrc522-0.0.7-py3-none-any.whl  
Requirement already satisfied: spidev in /usr/lib/python3/dist-packages (from mfrc522)  
Requirement already satisfied: RPi.GPIO in /usr/lib/python3/dist-packages (from mfrc522)  
Installing collected packages: mfrc522  
Successfully installed mfrc522-0.0.7  
pi@raspberrypi:~ $
```

Step 8: Write Python script which is used to write data from the RC522 to your RFID tags.

```
pi@raspberrypi: ~/rfiddemo
pi@raspberrypi:~ $ pwd
/home/pi
pi@raspberrypi:~ $ mkdir rfiddemo
pi@raspberrypi:~ $ cd rfiddemo/
pi@raspberrypi:~/rfiddemo $ pwd
/home/pi/rfiddemo
pi@raspberrypi:~/rfiddemo $ sudo nano write.py
```

```
GNU nano 2.7.4 File: write.py Modified
import RPi.GPIO as GPIO
from mfrc522 import SimpleMFRC522

reader = SimpleMFRC522()

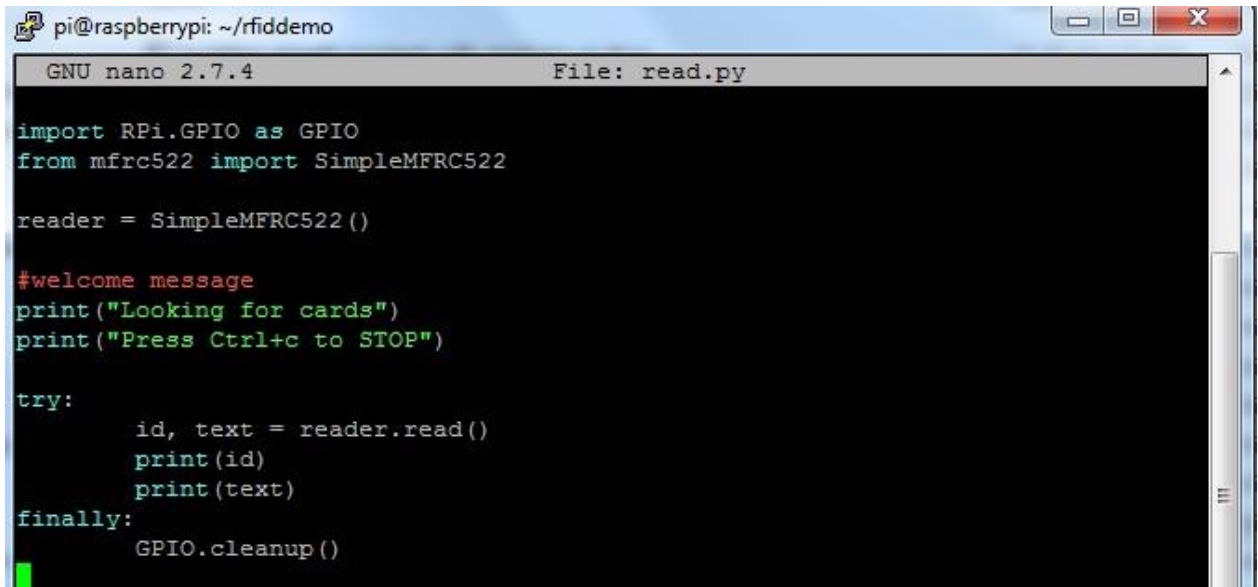
#welcome message
print("Looking for cards")
print("Press Ctrl+c to STOP")

try:
    text=input('Enter New Data:')
    print("Now place your tag to write.....")
    reader.write(text)
    print("Data Written Successfully")
finally:
    GPIO.cleanup()
```

Run above script

```
pi@raspberrypi:~ $ sudo nano writetest.py
pi@raspberrypi:~ $ sudo python writetest.py
Traceback (most recent call last):
  File "writetest.py", line 4, in <module>
    from mfrc522 import SimpleMFRC522
ImportError: No module named mfrc522
pi@raspberrypi:~ $ sudo python3 writetest.py
New data:welcome to IOT
Now place your tag to write
Written
pi@raspberrypi:~ $
```

Step 9: Write Python script which is used to read this data back off the RFID tag.

A screenshot of a terminal window on a Raspberry Pi. The window title is 'pi@raspberrypi: ~/rfiddemo'. The terminal shows the GNU nano 2.7.4 editor editing a file named 'read.py'. The code in the file is as follows:

```
import RPi.GPIO as GPIO
from mfrc522 import SimpleMFRC522

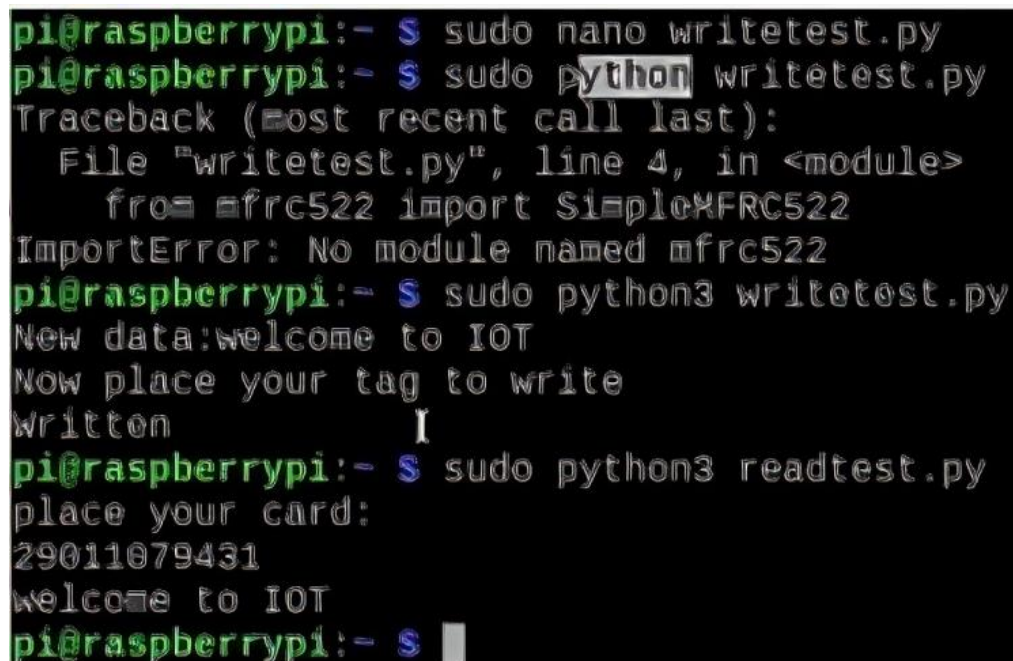
reader = SimpleMFRC522()

#welcome message
print("Looking for cards")
print("Press Ctrl+c to STOP")

try:
    id, text = reader.read()
    print(id)
    print(text)
finally:
    GPIO.cleanup()
```

Run above script

Output:

A screenshot of a terminal window showing the execution of two Python scripts. The first script, 'writetest.py', is run with 'sudo python writetest.py' and produces the output: 'New data:welcome to IOT' and 'Now place your tag to write'. The second script, 'readtest.py', is run with 'sudo python3 readtest.py' and produces the output: 'place your card:', '29011079431', and 'welcome to IOT'.

```
pi@raspberrypi:~$ sudo nano writetest.py
pi@raspberrypi:~$ sudo python writetest.py
Traceback (most recent call last):
  File "writetest.py", line 4, in <module>
    from mfrc522 import SimpleMFRC522
ImportError: No module named mfrc522
pi@raspberrypi:~$ sudo python3 writetest.py
New data:welcome to IOT
Now place your tag to write
Written
pi@raspberrypi:~$ sudo python3 readtest.py
place your card:
29011079431
welcome to IOT
pi@raspberrypi:~$
```