INDEX

SR. NO	AIM	DATE	SIGN
1.	Starting Raspberry pi OS, familiarising with Raspberry Pi Components and interface, connecting to ethernet, monitor, USB	02/07/25	
2.	Displaying different LED patterns with Raspberry Pi.	05/07/25	
3.	Displaying Time over 4-Digit 7-Segment Display using Raspberry Pi.	16/07/25	
4.	Controlling Raspberry Pi with Telegram App.	02/08/25	
5.	Interfacing Oscilloscope with Raspberry Pi.	06/08/25	
6.	Accessing RFID with Raspberry Pi.	09/08/25	
7.	Capturing image and video with PiCamera and Raspberry Pi.	23/08/25	
8.	GPS module interfacing with Raspberry Pi.	13/09/25	
9.	Fingerprint sensor interfacing with Raspberry Pi.	13/09/25	
10.	Displaying on LCD using Raspberry Pi	20/09/25	
11.	Setting up Wireless Access Point using Raspberry Pi.	20/09/25	

Practical 0:

Aim: Starting Raspberry pi OS, familiarising with Raspberry Pi Components and interface, connecting to ethernet, monitor, USB.

1. Familiarizing with Raspberry Pi Components and Interface

The Raspberry Pi is a compact, single-board computer with several key components and ports:

Component	**Function/Use**
ARM CPU/GPU	Main processor for running the OS and graphics
GPIO Pins	General Purpose Input/Output for connecting LEDs, sensors, etc.
HDMI Port	Connects to a monitor or TV for video output
USB Ports	Connects peripherals like keyboard, mouse, USB drives
Ethernet Port	For wired network connection (available on most models)
Audio Jack	Connects headphones or speakers
MicroSD Card Slot	Holds the OS and user files; required for booting
Power Port	Supplies power (typically via micro-USB or USB-C)
Camera/Display Ports	For official camera and display modules

2. Setting Up and Starting Raspberry Pi OS

a. Prepare the SD Card with Raspberry Pi OS

- Download and install Raspberry Pi Imager on your computer.
- Insert a microSD card (at least 8GB recommended) into your computer.
- Use Raspberry Pi Imager to select and write the latest Raspberry Pi OS onto the card.
- Once complete, safely eject the microSD card.

b. Assemble Your Raspberry Pi

- Insert the prepared microSD card into the Raspberry Pi's slot.
- Connect peripherals (monitor, keyboard, mouse, Ethernet cable if needed) while the Pi is powered off.
- Connect the power supply last. The Pi will power up and begin booting the OS.

3. Connecting to Peripherals

a. Connecting to a Monitor

- Use the appropriate HDMI cable (micro-HDMI for Pi 4/5, standard HDMI for Pi 3 and earlier).
- Plug the cable from the Pi's HDMI port to your monitor.
- Power on the monitor and select the correct HDMI input.

b. Connecting Keyboard and Mouse

- Plug a USB keyboard and mouse into the USB ports.
- For wireless devices, plug in their USB dongle receivers.

c. Connecting to Ethernet

- Plug an Ethernet cable into the Pi's Ethernet port and the other end into your router or switch.
- The Pi will automatically attempt to obtain an IP address via DHCP when powered on.

d. Connecting USB Devices

- Use the USB ports to connect USB drives, cameras, or other peripherals.
- If you need more ports, use a powered USB hub.

Tip: Always connect all peripherals before powering on the Raspberry Pi to avoid hardware detection issues.

Practical 1:

Aim: Displaying different LED patterns with Raspberry Pi.

Hardware Requirements

1. Raspberry Pi (any model with GPIO header)
2. MicroSD card with Raspberry Pi OS installed
3. Power supply for Raspberry Pi
4. Breadboard
5. LEDs (as many as you want to blink)
6. Resistors (one per LED, typically 220Ω–330Ω)
7. Jumper wires (male-to-female as needed)
8. Monitor, keyboard, mouse (for setup)
9. Optional: Multimeter (for checking connections)

Software Requirements

1. Raspberry Pi OS installed and running
2. Python 3
3. RPi.GPIO library (install with sudo apt-get install python3-rpi.gpio if not already present)

Circuit Connections:

LED	**Physical PIN Number**	**GPIO Number**
1	29	GPIO 05
2	31	GPIO 06
3	33	GPIO 13
4	35	GPIO 19

5	37	GPIO 26
All GND	39	GND

Source Code:

```
# Import the necessary libraries
import RPi.GPIO as GPIO
import time
# Set up GPIO using physical pin numbers
GPIO.setmode(GPIO.BOARD)
# List of GPIO pins where LEDs are connected
led_pins = [29, 31, 33, 35, 37]
# Set up each pin as an output and turn all LEDs off initially
for pin in led_pins:
    GPIO.setup(pin, GPIO.OUT)   # Set the pin as output
    GPIO.output(pin, GPIO.LOW)  # Ensure LED is OFF at start

# Define LED patterns: each sublist represents the ON/OFF state of all LEDs
patterns = [
    [1, 0, 0, 0, 0],  # Only first LED ON
    [0, 1, 1, 1, 0],  # Middle three LEDs ON
    [1, 1, 1, 1, 1],  # All LEDs ON
    [0, 1, 0, 1, 0],  # Alternate LEDs ON
    [0, 0, 1, 0, 0],  # Only center LED ON
]
try:
    while True:  # Loop forever
        for pattern in patterns:  # Go through each LED pattern
```

```
        # Set each LED according to the current pattern
        for pin, state in zip(led_pins, pattern):
            GPIO.output(pin, GPIO.HIGH if state else GPIO.LOW)  # Turn LED ON or OFF
        time.sleep(0.2)  # Wait 0.2 seconds before next pattern
except KeyboardInterrupt:
    print("Stopped by user.")  # Print message if program is stopped with Ctrl+C
finally:
    GPIO.cleanup()  # Reset all GPIO pins to a safe state.
```

Practical 2:

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

Hardware Requirements

1. Raspberry Pi
2. TM1637 4-Digit Seven Segment Display
3. MicroSD card with Raspberry Pi OS installed
4. Power supply for Raspberry Pi
5. Jumper wires (male-to-female as needed)
6. Monitor, keyboard, mouse (for setup)

Software Requirements

1. Raspberry Pi OS installed and running
2. Python 3
3. RPi.GPIO library

Circuit Connections:

TM1637 Board Pin	**Physical PIN Number**	**GPIO Number**
CLK	38	GPIO 20
DIO	40	GPIO 21
VCC	1	3.3 V
GND	6	GROUND

Command to download the library:

pip3 install raspberrypi-tm1637

Source Code:

```
import tm1637
from time import sleep

# Import thread module for running tasks in background.
# Python 2 uses thread, Python 3 renamed it to _thread.
try:
    import thread
except ImportError:
    import _thread as thread

# Initialize TM1637 display object with CLK on GPIO21 and DIO on GPIO20.
# Brightness range is from 0.0 (lowest) to 1.0 (highest).
Display = tm1637.TM1637(clk=21, dio=20, brightness=1.0)

try:
    print("Starting clock in the background (press CTRL+C to stop):")
    # Start the clock display in background thread.
    # military_time=True means 24-hour clock format.
    Display.StartClock(military_time=True)

    # Set the brightness to maximum (1.0).
    Display.SetBrightness(1.0)

    # Main loop to toggle the colon (double point) on the display every second
    while True:
        Display.ShowDoublepoint(True)  # Turn on colon/double point
        sleep(1)                       # Wait for 1 second
        Display.ShowDoublepoint(False) # Turn off colon/double point
```

```
        sleep(1)                    # Wait for 1 second


except KeyboardInterrupt:
    # If Ctrl+C is pressed, stop the clock and clean up the display before exiting.
    print("Properly closing the clock")
    Display.StopClock()
    Display.cleanup()
```

Clock_Display:

```
from time import sleep
import tm1637
# Initialize the TM1637 display
CLK = 21  # Clock pin
DIO = 20  # Data pin
display = tm1637.TM1637(CLK, DIO)


# Set display brightness (0-7)
display.brightness(1)
try:
    # Start the clock in the background (12-hour format)
    display.StartClock(military_time=False)


    while True:
        sleep(1)  # Keep the script running
except KeyboardInterrupt:
    # Stop the clock and clean up when CTRL + C is pressed
    display.StopClock()
    display.Clear()
    display.cleanup()
```

Extra:

```
import tm1637
from time import sleep
try:
        import thread
except ImportError:
    import _thread as thread

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.ShowDounlepoint(True)
     sleep(1)
   Display.ShowDounlepoint(False)
     sleep(1)



    Display.StopClock()
    thread.interrupt_main()
except KeyboardInterrupt:
    print("Properly closing the clock")
    Display.cleanup()
```

Practical 3:

Aim: Controlling Raspberry Pi with Telegram App.

Hardware Requirements

1. Raspberry Pi
2. MicroSD card with Raspberry Pi OS installed
3. Power supply for Raspberry Pi
4. Jumper wires (male-to-female as needed)
5. Monitor, keyboard, mouse (for setup)

Software Requirements

1. Raspberry Pi OS installed and running
2. Python 3
3. RPi.GPIO library

Circuit Connections:

Board Pin	**Physical PIN Number**	**GPIO Number**
GPIO	40	GPIO 21
VCC	1	3.3 V
GND	6	GROUND

Commands to download the library:

pip3 install telepot

pip3 RPi.GPIO

Source Code:

```
import time

import telepot

import RPi.GPIO as GPIO
```

```
from telepot.loop import MessageLoop

LED_PIN = 40  # Pin 40 = GPIO21
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
GPIO.setup(LED_PIN, GPIO.OUT)
GPIO.output(LED_PIN, 0)
def action(msg):
    chat_id = msg['chat']['id']
    command = msg['text']
    print('Got command:', command)

    if 'On' in command or 'on' in command:
        GPIO.output(LED_PIN, 1)
        bot.sendMessage(chat_id, "LED turned ON")
    elif 'Off' in command or 'off' in command:
        GPIO.output(LED_PIN, 0)
        bot.sendMessage(chat_id, "LED turned OFF")
    else:
        bot.sendMessage(chat_id, "Send 'on' or 'off' to control the LED.")

# Insert your bot token here
bot = telepot.Bot('BOT_TOKEN')
MessageLoop(bot, action).run_as_thread()
print('Listening for commands...')
while True:
    time.sleep(10)
```

Practical 4:

Aim: Displaying Oscilloscope with Raspberry Pi.

Hardware Requirements

1. Raspberry Pi
2. Oscilloscope ADS1115 ADC Module
3. MicroSD card with Raspberry Pi OS installed
4. Power supply for Raspberry Pi
5. Jumper wires (male-to-female as needed)
6. Monitor, keyboard, mouse (for setup)

Software Requirements

1. Raspberry Pi OS installed and running
2. Python 3

Circuit Connections:

Board Pin	**Physical PIN Number**	**GPIO Number**
SDA	3	GPIO 2
SLC	5	GPIO 3
VCC	2	5 V
GND	6	GROUND

Commands:

sudo raspi-config # for configuring interface I2C

reboot

sudo apt-get update

sudo apt-get upgrade

cd ~

sudo apt-get install build-essential python-dev python-smbus git

git clone https://github.com/adafruit/Adafruit_Python_ADS1x15.git

cd Adafruit_Python_ADS1x1z

sudo python setup.py install

cd examples

python simpletest.py

sudo apt-get install python-matplotlib

sudo apt-get install python-pip

sudo pip install drawnow

Source Code:

```
import time
import matplotlib.pyplot as plt
import Adafruit_ADS1x15

# Create ADS1115 ADC (16-bit)
adc = Adafruit_ADS1x15.ADS1115()
GAIN = 1

values = []
max_points = 50

plt.ion()
fig, ax = plt.subplots()
line, = ax.plot([], [], 'ro-')
ax.set_ylim(-5000, 5000)
ax.set_title('ADS1115 Oscilloscope')
ax.set_ylabel('ADC Value')

def update_plot():
```

```
    line.set_data(range(len(values)), values)
    ax.set_xlim(0, max_points)
    fig.canvas.draw()
    fig.canvas.flush_events()

print('Reading ADS1115 channel 0')

try:
    while True:
        value = adc.read_adc(0, gain=GAIN)
        print('Channel 0: {}'.format(value))
        values.append(value)
        if len(values) > max_points:
            values.pop(0)
        update_plot()
        time.sleep(0.1)
except KeyboardInterrupt:
    print('Exiting program')
    plt.ioff()
    plt.show()
```

Practical 5:

Aim: Accessing RFID with Raspberry Pi.

Hardware Requirements

1. Raspberry Pi
2. RFID EM 18 Module
3. MicroSD card with Raspberry Pi OS installed
4. Power supply for Raspberry Pi
5. Jumper wires (male-to-female as needed)
6. Monitor, keyboard, mouse (for setup)

Software Requirements

1. Raspberry Pi OS installed and running
2. Python 3

Circuit Connections:

Board Pin	**Physical PIN Number**	**GPIO Number**
TX	10	GPIO 15
VCC	2	5 V
GND	6	GROUND

Commands:

(disable login shell over serial, enable serial hardware)

sudo raspi-config - # for configuring interface I2C

reboot

sudo apt-get update

sudo apt-get upgrade

pip3 install pyserial

Source Code:

```
import serial
import time
# Configure serial port (UART)
ser = serial.Serial('/dev/serial0', 9600, timeout=1)
# Replace with your card's actual ID after reading it once
AUTHORIZED_CARD = "enter_your_card_no"
print("RFID Access System Ready")
print("Tap your card...")

try:
    while True:
        data = ser.read(12)  # EM-18 sends 12 characters per tag
        if data:
            card_id = data.decode('utf-8').strip()
            print(f"Card Detected: {card_id}")

            if card_id == AUTHORIZED_CARD:
                print(" Access Granted — Hello!")
            else:
                print(" Access Denied — Nah!")

            time.sleep(1)  # Small delay before next read
except KeyboardInterrupt:
    print("\nExiting program...")
finally:
    ser.close()
```

To find your card_Id:

```
import serial
```

```
ser = serial.Serial('/dev/serial0', 9600, timeout=1)
print("Scan your card...")
while True:
    data = ser.read(12)
    if data:
        print(data.decode('utf-8').strip())
```

Practical 6:

Aim: Capturing image and video with PiCamera and Raspberry Pi.

Hardware Requirements

1. Raspberry Pi
2. Picamera
3. MicroSD card with Raspberry Pi OS installed
4. Power supply for Raspberry Pi
5. Monitor, keyboard, mouse (for setup)

Software Requirements

1. Raspberry Pi OS installed and running
2. Python 3

Commands:

sudo raspi-config - # for configuring interface I2C

Interface -> enable Camera

reboot

sudo apt-get update

sudo apt-get upgrade

sudo apt install python3-picamera

Source Code For Image:

```
from picamera import PiCamera
import datetime
import os


# Create directory if not exists
image_folder = '/home/pi/images'
os.makedirs(image_folder, exist_ok=True)


camera = PiCamera()
```

```
camera.resolution = (1024, 768)  # Optional: Set resolution

timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
#image_path = f"{image_folder}/image_{timestamp}.jpg"
image_path = "{}/image_{}.jpg".format(image_folder, timestamp)

camera.start_preview()  # Optional: Shows camera preview
camera.capture(image_path)
camera.stop_preview()

#print(f"Image captured and saved to {image_path}")
print("Image captured and saved to {}".format(image_path))
```

Source Code For Video:

```
from picamera import PiCamera
from time import sleep
import datetime
import os
import subprocess

# Create directory if not exists
video_folder = '/home/pi/videos'
os.makedirs(video_folder, exist_ok=True)

camera = PiCamera()
camera.resolution = (1024, 768)  # Optional: Set resolution

timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
```

```
video_h264_path = "{}/video_{}.h264".format(video_folder, timestamp)
video_mp4_path = "{}/video_{}.mp4".format(video_folder, timestamp)

camera.start_preview()  # Optional: Shows camera preview
camera.start_recording(video_h264_path)
sleep(10)  # Records for 10 seconds (change as needed)
camera.stop_recording()
camera.stop_preview()

print("Raw video recorded and saved to {}".format(video_h264_path))

# Convert .h264 to .mp4 using MP4Box (install with sudo apt install gpac)
convert_command = ["MP4Box", "-add", video_h264_path, video_mp4_path]

try:
    subprocess.run(convert_command, check=True)
    print("Converted to MP4: {}".format(video_mp4_path))
    # Optional: Remove raw .h264 file if conversion successful
    os.remove(video_h264_path)
except Exception as e:
    print("Failed to convert video:", e)
```

Practical 7:

Aim: GPS module interfacing with Raspberry Pi.

Hardware Requirements

1. Raspberry Pi
2. GPS Module
3. MicroSD card with Raspberry Pi OS installed
4. Power supply for Raspberry Pi
5. Jumper wires (male-to-female as needed)
6. Monitor, keyboard, mouse (for setup)

Software Requirements

1. Raspberry Pi OS installed and running
2. Python 3

Circuit Connections:

Board Pin	**USB TO TTL**
TX	RX
RX	TX
VCC	5 V
GND	GND

Commands:

sudo apt-get update

sudo apt-get upgrade

sudo pip3 install pyserial

pip3 install gps3

sudo apt install python-gps

sudo apt install gpsd gpsd-clients

```
sudo apt install gpsd gpsd-clients python3-gps
#sudo systemctl stop gpsd.socket
#sudo gpsd /dev/serial0 -F /var/run/gpsd.sock
```

Source Code:

```
import serial
def parse_GPGGA(sentence):
    try:
        parts = sentence.split(',')
        if parts[0] == "$GPGGA" and len(parts) > 5:
            lat = parts[1]
            lat_dir = parts[2]
            lon = parts[3]
            lon_dir = parts[4]
            if lat and lon and lat_dir and lon_dir:
                lat_deg = int(float(lat) / 100)
                lat_min = float(lat) - lat_deg * 100
                latitude = lat_deg + lat_min / 60
                if lat_dir == 'S':
                    latitude = -latitude
                lon_deg = int(float(lon) / 100)
                lon_min = float(lon) - lon_deg * 100
                longitude = lon_deg + lon_min / 60
                if lon_dir == 'W':
                    longitude = -longitude
                return latitude, longitude
    except Exception:
        pass
    return None, None
```

```
# Initialize serial connection here!
ser = serial.Serial('/dev/ttyUSB0', baudrate=9600, timeout=1)


while True:
    line = ser.readline()
    try:
        line = line.decode('ascii', errors='replace')
    except AttributeError:
        pass
    print("Received line:", line.strip())
    if line.startswith('$GPGGA'):
        lat, lon = parse_GPGGA(line)
        if lat and lon:
            print("Latitude: %.6f, Longitude: %.6f" % (lat, lon))
        else:
            print("Invalid or no GPS fix yet.")
```

Practical 8:

Aim: Fingerprint sensor interfacing with Raspberry Pi.

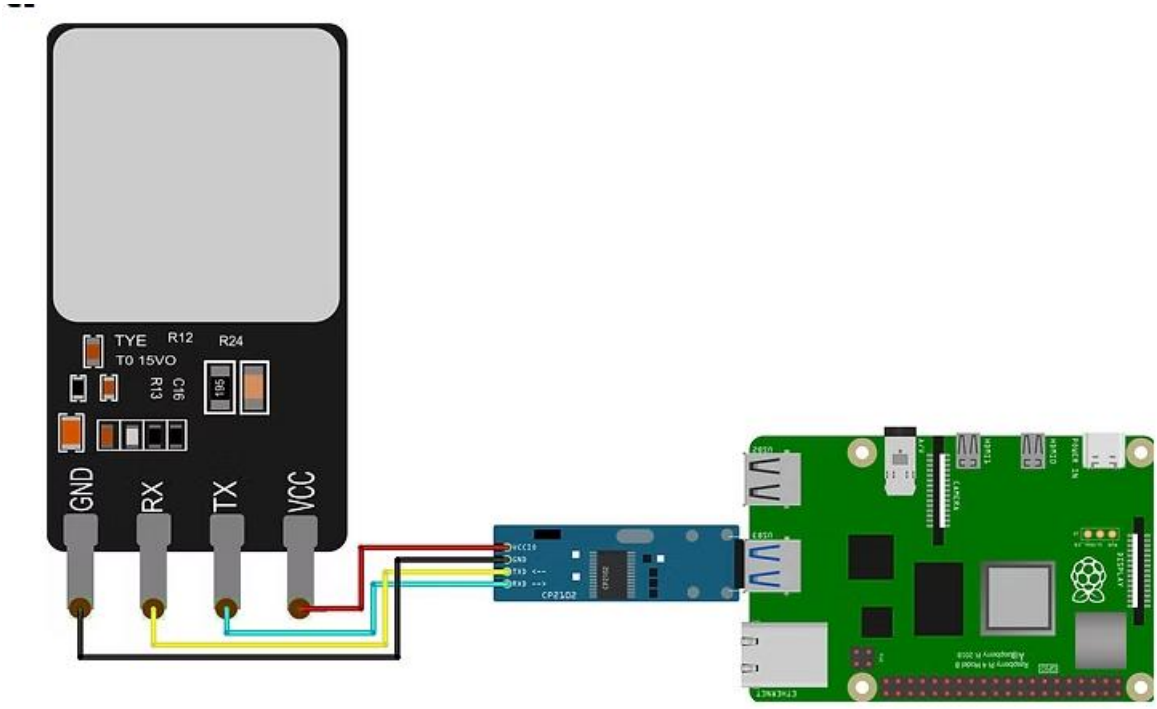
Hardware Requirements

1. Raspberry Pi
2. Fingerprint module
3. MicroSD card with Raspberry Pi OS installed
4. Power supply for Raspberry Pi
5. Jumper wires (male-to-female as needed)
6. Monitor, keyboard, mouse (for setup)

Software Requirements

1. Raspberry Pi OS installed and running
2. Python 3

Circuit Connections:



Board Pin	**USB TO TTL**
TX	RX

RX	TX
VCC	VCC
GND	GND

Commands:

sudo apt-get update

sudo apt-get upgrade

sudo pip3 install pyfingerprint

sudo pip3 install adafruit-circuitpython-fingerprint pyfingerprint

git clone https://github.com/adafruit/Adafruit_CircuitPython_Fingerprint.git

git clone https://github.com/bastianraschke/pyfingerprint.git

Source Code:

```
from pyfingerprint.pyfingerprint import PyFingerprint
import time


try:
    f = PyFingerprint('/dev/ttyUSB0', 57600, 0xFFFFFFFF, 0x00000000)
    if not f.verifyPassword():
        raise ValueError('The given fingerprint sensor password is wrong!')
except Exception as e:
    print('Fingerprint sensor initialization failed:', e)
    exit(1)


def enroll():
    print('Place finger to enroll...')
    while f.readImage() == False:
        pass
    f.convertImage(0x01)
```

```
    result = f.searchTemplate()
    if result[0] >= 0:
        print('Finger already enrolled at position #' + str(result[0]))
        return
    print('Remove finger...')
    time.sleep(2)
    print('Place same finger again...')
    while f.readImage() == False:
        pass
    f.convertImage(0x02)
    if f.compareCharacteristics() == 0:
        print('Fingers do not match')
        return
    f.createTemplate()
    position = f.storeTemplate()
    print('Finger enrolled at position #' + str(position))

def search():
    print('Place finger to search...')
    while f.readImage() == False:
        pass
    f.convertImage(0x01)
    result = f.searchTemplate()
    position = result[0]
    if position >= 0:
        print('Finger found at position #' + str(position))
    else:
        print('Finger not found')
```

```
print('Choose e to enroll or s to search:')
choice = input("> ").lower()
if choice == 'e':
    enroll()
elif choice == 's':
    search()
else:
    print('Invalid choice')
```

Practical 9:

Aim: Displaying on LCD using Raspberry Pi.

Hardware Requirements

1. Raspberry Pi
2. LCD module
3. MicroSD card with Raspberry Pi OS installed
4. Power supply for Raspberry Pi
5. Jumper wires (male-to-female as needed)
6. Monitor, keyboard, mouse (for setup)

Software Requirements

1. Raspberry Pi OS installed and running
2. Python 3

Circuit Connections:

Board Pin	**Physical PIN Number**	**GPIO Number**
SCL	5	GPIO 5
SDA	3	GPIO 3
VCC	2	5 V
GND	6	GROUND

Commands:

sudo apt-get update

sudo apt-get upgrade

sudo pip3 install RPLCD

sudo apt install python3-smbus i2c-tools

Find your LCD's I2C address (often 0x27 or 0x3f):

sudo i2cdetect -y 1

Source Code:

```
from RPLCD.i2c import CharLCD
import time
# Initialize the LCD, change the address if your I2C address is different
lcd = CharLCD('PCF8574', 0x27, port=1, cols=16, rows=2)


try:
    lcd.write_string('Hello, World!')
    time.sleep(2)
    lcd.clear()
    lcd.write_string('Raspberry Pi\nLCD Demo')
    time.sleep(3)
finally:
    lcd.clear()
```

Practical 10:

Aim: Setting up Wireless Access Point using Raspberry Pi.

Hardware Requirements

1. Raspberry Pi
2. MicroSD card with Raspberry Pi OS installed
3. Power supply for Raspberry Pi
4. Jumper wires (male-to-female as needed)
5. Monitor, keyboard, mouse (for setup)

Software Requirements

1. Raspberry Pi OS installed and running

Steps:

1. Update and Install Required Packages

Open terminal and run:

sudo apt update

sudo apt upgrade -y

sudo apt install hostapd dnsmasq

3. Configure a Static IP for wlan0

Edit the DHCP client configuration:

sudo nano /etc/dhcpcd.conf

Add these lines at the end:

interface wlan0

static ip_address=192.168.4.1/24

nohook wpa_supplicant

Save and exit (Ctrl+X, Y, Enter).

Restart the DHCP client:

sudo service dhcpcd restart

4. Configure DNSmasq (DHCP Server)

Backup and edit dnsmasq config:

sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig

sudo nano /etc/dnsmasq.conf

Add this configuration:

interface=wlan0 # Use interface wlan0

dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h

Save and exit.

5. Configure Hostapd (Access Point)

Create hostapd config file:

sudo nano /etc/hostapd/hostapd.conf

Add this configuration:

interface=wlan0

driver=nl80211

ssid=YourSSIDName

hw_mode=g

channel=7

wmm_enabled=0

macaddr_acl=0

auth_algs=1

ignore_broadcast_ssid=0

wpa=2

wpa_passphrase=YourPassword

wpa_key_mgmt=WPA-PSK

rsn_pairwise=CCMP

Replace YourSSIDName and YourPassword with your desired Wi-Fi name and password.

Edit /etc/default/hostapd to point to the config file:

sudo nano /etc/default/hostapd

Find and modify:

DAEMON_CONF="/etc/hostapd/hostapd.conf"

Save and exit.

6. Enable IP Forwarding

Edit sysctl config:

sudo nano /etc/sysctl.conf

Uncomment or add this line:

net.ipv4.ip_forward=1

Apply changes now:

sudo sysctl -p

7. Setup NAT with iptables

sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"

Edit /etc/rc.local to load iptables at boot:

sudo nano /etc/rc.local

Add above the exit 0 line:

iptables-restore < /etc/iptables.ipv4.nat

Save and exit.

8. Start Services and Enable at Boot

sudo systemctl unmask hostapd

sudo systemctl enable hostapd

sudo systemctl enable dnsmasq

sudo systemctl start hostapd

sudo systemctl start dnsmasq

9. Reboot and Test

Reboot your Raspberry Pi:

bash

sudo reboot