Spdt PRACTICAL

PAGE No.: _____

DATE : __/__/__

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

1) Starting Raspbian OS:

- Raspbian OS is the official operating System for Raspberry Pi
- To get started, You'll need a microsd card with Raspbian installed, a power supply, and access to Peripherals like a monitor, Keyboard and mouse.
- Once powered on, the Raspberry Pi will boot into Raspbian OS, allowing you to interact with the desktop Environment.

2) Familiarizing with Raspberry Pi Components & Interface.

The Raspberry Pi is a Compact Computer with various Components such as:-

- GPIO {General Purpose I/O} pins for Connecting hardware
- HDMI port for Connecting a monitor or display.
- Ethernet port for wired internet Connectivity
- USB Ports for Connecting Peripherals like keyboard, mouse and external Storage.

PAGE No.: _____

Spdt

DATE : __/__/__

3) Connecting to Ethernet

- To Connect to the internet via Ethernet, plug an Ethernet Cable into the Raspberry Pi's Ethernet Port
- If the router provides DHCP, the Raspberry Pi will automatically obtain an IP address and Connect to the network.

4) Connecting to Monitor

- use the HDMI Port to Connect the Raspberry Pi to a monitor or TV. Ensure that the monitor is set to the Correct input Source.

5) Connecting USB Devices.

- The Raspberry Pi has multiple USB ports for Connecting peripherals like a Keyboard, mouse and External Storage devices.
- Simply plug in the devices, and they should be recognized by the system automatically.

Spdt PRACTICAL

PAGE No.: _____

DATE : __/__/__

Aim:- Displaying different LED patterns with Raspberry Pi

Hardware Requirements

The following hardware Components are required for this project :-

→ Raspberry Pi (any model with GPIO Pins)
→ LEDs (multiple for different Patterns)
→ Resistors (typically 330 Ω or 220 Ω for Current limiting)
→ Jumper Wires (for Connecting the LEDs to GPIO Pins.)
→ Bread board (for easy Connection of LED's and resistors)
→ Power Supply (for Raspberry Pi)

Software Requirements

• To run the necessary scripts and Control the LEDs, the fellowing Software is required.

→ Raspbian OS {pre-installed on the Raspberry Pi}
→ Python Programming languages {comes pre-installed on Raspbian}
→ GPIO zero Library {to simplify the process of Controlling GPIO pins}
→ Any text Editor or IDE for writing Python Code {eg VScode}

PAGE No.: _____

DATE : __/__/__

Spdt

Code:

```
import time
import RPI. GPIO as GPIO
GPIO.Setmode (GPIO.Board)
GPIO. Setwarnings (False)

LED_RED = 7
LED_Yellow = 11

GPIO. Setup (LED_RED, GPIO. OUT)
GPIO. Setup (LED_Yellow, GPIO. OUT)

while 1:
    GPIO. output (LED_RED, True)
    time. sleep (1)
    GPIO. output (LED_Yellow, True)   time. sleep(1)
    GPIO. output (LED_Red, False)
    time. sleep (1)
    GPIO. Output (LED_Yellow, False)
    time Sleep (1)
```

Spdt PRACTICAL

PAGE No.: _____

DATE : __/__/__

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

Abstract:- This project deals with Interfacing of 4-digit Seven Segment Display Module with Raspberry Pi and display Time over it. only two data wires are required for I2C – a data line (SDA) and a clock line (SCL)

Required Components:-

1) Raspberry Pi 3
2) Power Supply 12V/2Amp
3) USB Keywords
4) USB Mouse
5) Micro SD card
6) 4-Digit 7-Segment Display module
7) Discover Board

Procedure:-

Step 1:- Connection of Discover Development Board
Step 2:- Connection of Raspberry PI
Step 3:- Switch ON power supply.
Step 4:- Login to Raspberry pi Terminal
Step 5:- Create a newfile with an extension .py
Step 6:- Open the file with Python 2 IDLE only
Step 7:- Type and run the program and see the output on 4-Digit 7 Segment display board.

PAGE No.: _____

DATE : __/__/__

Spdt

Code :-

```
import time
import Datetime
from lib import tm1637 as obj

Display = obj.TM1637(2,3,5)
Display.clear()

while (True):
    now = datetime.datetime.now()
    hour = now.hour
    minute = now.minute
    Second = now.second
    Display.clear()
    val = [(int(hour/10)), (hour%10), (int(minute/10)), (minute%10)]
    Display.show(val)
    Display.show(val)
    Display.showDoublepoint((Second%2))
    time.sleep(0.25)
```

Conclusion

Thus we have Studied and displayed Time over 4-Digit 7-Segment Display using Raspberry Pi.

Spdt PRACTICAL

PAGE No.: _____

DATE : __/__/__

Aim: Raspberry Pi based Oscilloscope

Abstract: The Oscilloscope is an electronic test instrument that allows the visualization and observation of varying Signal voltages usually as a two dimensional plot with one or more signals plotted against time. Today's project will Aptseek to replicate the Signal visualization Capabilities of the oscilloscope using the Raspberry Pi and an analog to digital Converter module.

Required Components:-

1) Raspberry Pi 3
2) Power Supply 12V/2Amp
3) USB Keyboard
4) USB Mouse
5) Micro SD Card
6) ADS1115ADC
7) Discover Board
8) Analog Input As per Availability

Procedure:-

Step 1 :- Connection of Discover Development Board
Step 2 :- Connection of Raspberry Pi.
Step 3 :- Switch ON power supply
Step 4 :- Login to Raspberry Pi Terminal
Step 5 :- Create a new file with an extension.py
Step 6 :- Open the file with python 2 IDLE only
Step 7 :- Type and run the program and see the output on terminal

PAGE No.: _____

DATE : __/__/__

Spdt

Code

```
import time
import matplotlib.pyplot as plt
from drawnow import *
import Adafruit_ADS1x15
adc = Adafruit_ADS1x15.ADS1115()
GAIN = 1
val = []
cnt = 0
plt.ion()
# Start Continous ADC conversion on channel 0 using the
  Previous gain value
adc.start_adc(0, gain = GAIN)
print('Reading ADS1x15 channel 0')
# Create the figure function
def makeFig():
plt.ylim(-5000, 5000)
plt.title('USIT_IOT Lab Oscilloscope')
plt.grid(True)
plt.ylabel('ADC Outputs')
plt.plot(val, 'ro-', label='Channel 0')
plt.legend(loc='lower right')
```

Spdt

PAGE No.: _____

DATE : __/__/__

```
while (true):
    #Read the last ADC conversion value and print
    value = adc.get_last_result()
    print('Channel 0: {0}'.format(value))
    #Sleep for half a second
    time.sleep(0.1)
    val.append(int(value))
    drawnow(makeFig)
    plt.pause(.000001)
    cnt = cnt+1
    if(cnt>50):
        val.pop(0)
```

Conclusion:-

Thus Raspberry Pi Based Oscilloscope has been studied and implemented

Spdt PRACTICAL

PAGE No.: _____

DATE : / /

Aim:- RFID interfacing with Raspberry Pi.

Abstract:- Radio-frequency identification (RFID) uses electromagnetic fields to automatically identify and track tag attached to objects. The Tags contain electronically stored information. Using RFID the exchange of data between tags and reader is rapid automatic and does not require direct contact or line of sight. By employing RFID, much secured entry systems can be developed without incurring huge costs.

Required Components

1) Raspberry Pi 3
2) Power Supply 12V / 2Amp
3) USB Keyboard
4) USB Mouse
5) Micro SD Card
6) Discover Board
7) RFID Reader
8) 16×2 LCD.

Procedure:-

Step1:- Connection of Discover Development Board
Step2:- Connection of Raspberry Pi
Step3:- Switch ON Power Supply
Step4:- Login to Raspberry Pi Terminal
Step5:- Create a new file with an extension.py
Step6:- Create a new file with an extension.py
Step7:- Open the file with Python 2 IDLE only.

Step8:- Type and run the program and follow the steps shown on Terminal window, Show RFID Card See the output on Terminal and LCD module.

Code:-

```
import time
import RPi.GPIO as GPIO
import sys
from lib import SimpleMFRC522
~~~~ GPIO Configuration ~~~~
GPIO.setwarnings(False)
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)
~~~~ Global Variables ~~~~
~~~~ Lcd Defines ~~~~
#Define GPIO to LCD Mapping
LCD_RS = 13
LCD_E = 19
LCD_D4 = 6
LCD_D5 = 5
LCD_D6 = 21
LCD_D7 = 26
#Configure GPIO as output
GPIO.setup(LCD_E, GPIO.OUT) #E
GPIO.setup(LCD_RS, GPIO.OUT) #RS
GPIO.setup(LCD_D4, GPIO.OUT)# DB4
GPIO.setup(LCD_D5, GPIO.OUT) # DB5
GPIO.setup(LCD_D6, GPIO.OUT) # DB6
GPIO.setup(LCD_D7. GPI.OUT) #DB7

#Define Some device Constants
LCD_WIDTH = 16 # maximum character per line
LCD_CHR = True
```

PAGE No.:

DATE : / /

Spdt

```
LCD_CMD = False
LCD_Line1 = 0x80 # Ram address for 1st line
LCD_Line2 = 0xC0 # Ram address for 2nd line
# Timing Constants
E_PULSE = 0.0005
E_DELAY = 0.0005
LCD_SCRL_DEL = 0.14529 #LCD Scrolling delay
# LCD function to initialize display character on it
def lcd_init()
    # initialize Display
    lcd_byte(0x33, LCD_CMD) # 110011 initialise
    lcd_byte(0x32, LCD_CMD): # 110010 Initialise
    lcd_byte(0x06, LCD_CMD) # 000110 Cursor move direction
    lcd_byte(0x0C, LCD_CMD) # 001100 Display on,cursor off,blink off
    lcd_byte(0x28, LCD_CMD) # 101000 Data length,number of line,font size
    lcd_byte(0x01, LCD_CMD) # clear display
    time.sleep(E_DELAY)
def lcd_byte(bits, mode):
        Send byte to data pins
    bits = data
    mode = True for character or False for Command
GPIO.output(LCD_RS, mode) #RS
    High bits.
GPIO.output(LCD_D4, False)
GPIO.output(LCD_D5, False)
GPIO.output(LCD_D6, False)
GPIO.output(LCD_D7, False)
```

```
    if bits&0x10==0x10:
        GPIO.output(LCD_D4,True)
    if bits&0x20==0x20:
        GPIO.output(LCD_D5,True)
    if bits&0x40==0x40:
        GPIO.output(LCD_D6,True)
    if bits&0x80==0x80:
        GPIO.output(LCD_D7,True)
# Toggle 'Enable' pin
lcd_toggle_enable()
# low bits
GPIO.output(LCD_D4,False)
GPIO.output(LCD_D5,False)
GPIO.output(LCD_D6,False)
GPIO.output(LCD_D7,False)
if bits&0x01==0x01:
    GPIO.output(LCD_D4,True)
if bits&0x02==0x02:
    GPIO.output(LCD_D5,True)
if bits&0x04==0x04:
    GPIO.output(LCD_D6,True)
if bits&0x08==0x08:
    GPIO.output(LCD_D7,True)
# Toggle 'Enable' pin
lcd_toggle_enable()
def lcd_toggle_enable():
# Toggle enable
time.sleep(E_DELAY)
GPIO.output(LCD_E,True)
time.sleep(E_PULSE)
GPIO.output(LCD_E,False)
time.sleep(E_DELAY)
```

Spdt

PAGE No.: _____

DATE : __/__/__

```
def lcd_string(message,line):
# Send string to display
message = message.ljust(LCD_WIDTH," ")
lcd_byte(line, LCD_CMD)
for i in range(LCD_WIDTH):
lcd_byte(ord(message[i]),LCD_CHR)
def lcd_string_scroll(message, line, SCL_DELAY):
    string = message.strip()
    string = "  " + string
    string "length = " + str(len(string))
    if line=1:
        lcd_string("  ", line)
        for i in range(len(message)+10+16):
                lcd_string(string, line)
                time.sleep(SCL_DELAY)
                string = string[1:]+"  "
        pass

lcd_string("  ", line)
for i in range(len(message)+1+16):
        lcd_string(string, line)
        time.sleep(SCL_DELAY)
        string = string[1:]+"  "
pass
```

~~~~ End of lcd defines ~~~~

~~~~ User function ~~~~

```
def delaymS (time_in_msec):
    time.sleep((time_in_msec/1000)):
    pass
def delaySec (time_in_Sec):
    time.Sleep (time_in_sec)
def helpMe():
    print "\n Help documentation for R30S finger print module use on PI "
    print "\n\n \n\n\n\n Press any key to get out from here"
    exi = raw_input()
    print "\n"
    pass
def clearScreen():
    for i in range (50):
        print "\n"
        print pass
        pass
def clearScreen (no_of_line):
    for i in range (no_of_line):
        print "\n"
        pass
        pass
def writeToRFIDTag():
    reader = SimpleMFRC522.SimpleMFRC522()
try:
    text = raw_input ("Enter new ID for your TAG :")
    print ("Place your RFID TAG on area of RFID reader module ")
    reader.write(text)
    print ("Writing RFID TAG done")
```

Spdt

PAGE NO.:

DATE: / /

```
except Exception as e:
    print "Writing RFID TAG fail"
    print "error:" + Str(e)
pass
def readFromRFIDTag():
reader = SimpleMFRC522.SimpleMFRC522()
print ("Place Your RFID Tag on area of RFID reader module")
id, text = reader.read()
print (id)
print (text)
except Exception as e:
    print "Error reading RFID Tag"
    print "error" + Str(e)
pass
def deleteFromRFIDTag():
  reader = SimpleMFRC522.SimpleMFRC522()
try:
  text = ""
  print ("Place Your RFID TAG on Area of RFID Reader module...")
  reader.write(text)
  print ("Deleting ID from RFID TAG done")
except Exception as e
    print "Deleting ID from RFID TAG fail"
    print "error:" + Str(e)
    pass
    pass
## Tries to initialize the user
    ~~~ End of User Function ~~~
```

```
# Thread Function for LCD
def test_on_LCD():
    print "17 here\n"
    lcd_init()    # initialize 16x2 LCD Module
    try:
        lcd_string("Keep your RFID ", LCD_LINE_1)
        lcd_string("TAG on reader...", LCD_LINE_2)
        reader = SimpleMFRC522.SimpleMFRC522()
        id, text = reader.read()
        print(id)
        print(text)
        lcd_string("ID", LCD_LINE_1)
        lcd_string(str(id)+" ", LCD_LINE_2)
        delay_sec(1)
        lcd_string("Value", LCD_LINE_1)
        lcd_string(str(text)+" ", LCD_LINE_2)
        delay_sec(2)
    except Exception as e:
        lcd_string("Failure reading", LCD_LINE_1)
        lcd_string("RFID module", LCD_LINE_2)
        print("Error: "+str(e))
lcd_string("Choose option in", LCD_LINE_1)
lcd_string("Menu...", LCD_LINE_2)

def main():
clearScreen(25)
choice = 0   # for choosing option
global fingerprint_module  # object for fingerprint module
```

Spdt

PAGE No.: _____

DATE : __ / __ / ____

```
print "Initialization done... Starting Program"
delay_sec(1)

while True:  # 1st loop
    clearScreen(25)
    print" Program to Demonstrate use of RFID-
          RC522 module in LCD"
    print"\n\n Choose option in the menu below.
          \n 1) write RFID Tag
          \n2) Read RFID Tag.
          \n3) Clear RFID
          \n4) RFID on LCD
          \n5) Help
          \n6) Exit Program. \n\n"
    print ="Enter choice"
    choice = int(raw_input())

# Enroll finger into Database
if choice = 1:
    writeToRFID_Tag()
    pass
# Search finger into Database
if choice = 2:
    read_from_RFID_Tag()
```

```
# Delet from Database
if choice == 3
deleteFromRFIDTag()
pass
# Run finger Print with LCD
if choice == 4:
    testOnLCD()
pass
# Help case with documentation of using this module
if choice == 5:
    helpMe()
    pass

if choice == 6:
    exit()
    pass

delaySec(3)
pass                    # end of 1st loop
pass

if name == 'main':
    try:
        main()
    except KeyboardInterrupt:
        pass
```

PAGE No.: _____

DATE : / /

Spdt

finally:

```
lcd_String("Program ", LCD_Line_1)
lcd_String("Terminated...", LCD_Line_2)
GPIO.cleanUP()
print "\n\n Program Terminated.
```

Conclusion

Thus we have studied and RFID interfaced with Raspberry Pi.

Spdt

PRACTICAL

PAGE No.: _____

DATE : __/__/__

Aim: Visitor Monitoring with Raspberry Pi and Pi Camera.

Abstract:- This Project Deals with interfacing Pi Camera with Raspberry Pi to Capture the image of every visitor which has entered through the Gate or door, whenever any person is arrived at the Gate, he/She has to press a button to open the Gate, and as Soon as he/she press the button, his/her picture will be Captured and Saved in the System with the Date and time of the entry. This can be very useful for Security and Surveillance purpose. This system is useful in offices or factories where visitor entry record is need to be maintained.

Required Components.

1) Raspberry Pi3
2) Power Supply 12V/2Amp
3) USB Keyboard
4) USB Mouse
5) Micro SD card
6) Discover Board
7) Pi Camera with CSI Connector.

Procedures:-

Step1:- Connection of Discover Development Board.
Step2:- Connection of RaspberryPi
Step3:- Switch ON Power Supply
Step4:- Login to Raspberry Pi Terminal
Step5:- Install Camera Library
Step6:- Create a new file with extension.Py
Step7:- Open the file with Python 2 IDLE only
Step8:- Type and Run the Program and See the output on terminal

PAGE No.: _____

DATE : __/__/__

Spdt

Code :-

```
from pib.tempimage import TempImage
from picamera.array import PiRGBArray
from picamera import PiCamera.
import argparse
import warnings
import datetime
import dropbox
import imutils
import json
import time
import cv2
# filter warnings load the configuration and initialize the Dropbox
#client
warnings.filterwarnings("ignore")
Conf = json.load(open(args["conf"]))
Conf = json.load(open('conf.json'))
client = None
#check to see if the Dropbox should be used.
if Conf["use_dropbox"]:
#Connect to dropbox and start the session authorization process
Client = dropbox.Dropbox(Conf["dropbox_access_token"])
print("[SUCCESS] dropbox account linked")
#initialize the camera and Grab a reference to the raw camera Capture
Camera = PiCamera()
Camera.resolution = typle(Conf["resolution"])
Camera.framerate = Conf["fps"]
rawCapture = PiRGBArray(Camera,size=typle(Conf["resolution"]))
```

```
# increment the motion counter
motionCounter += 1
# check to see if the number of frames with consistent motion is
# high enough
if motionCounter >= conf["min_motion_frames"]:
# check to see if dropbox should be used
if conf["use_dropbox"]:
# write the image to temporary file
t = TempImage()
cv2.imwrite(t.path, frame)
# upload the image to Dropbox and cleanup the tempory image
print("[UPLOAD] {}".format(ts))
path = "{base_path}/{timestamp}.jpg".format(
    base_path=conf["dropbox_base_path"], timestamp=ts)
client.files_upload(open(t.path, "rb").read(), path)
t.cleanup()
# update the last uploaded timestamp and reset the motion
# counter
lastUploaded = timestamp
motionCounter = 0
# otherwise, the room is not occupied
else:
motionCounter = 0
# check to see if the frames should be displayed to screen
if conf["show_video"]:
# display the security feed
cv2.imshow("Security Feed", frame)
key = cv2.waitKey(1) & 0xFF
```

```
# if the 'q' key is pressed, break from the loop
if (key == ord("q")):
break
# clear the stream in preparation for the next frame
raw.capture.truncate(0)
```

Conclusion

Thus we have studied and implemented visitor monitoring with Raspberry Pi and Pi camera.

Spdt PRACTICAL

PAGE No.: _____

DATE : __/__/__

Aim:- Controlling Raspberry Pi with whatsapp.

Abstract:- This Demonstrates Controlling an LED Connected to a Raspberrypi using whatsApp messages through the Yowsup Library. By Configuring Yowsup to Communicate with whatsapp users can send messages to the Raspberry pi to turn the LED on or off.

Required Components:-

1) RaspberryPi
2) LED
3) Resistor
4) Breadboard and jumper wires
5) Yowsup library for whatsApp messaging
6) GPIO pytho library.

Steps for installing Yowsup Library

1) Install Yowsup Library by cloning it from github.
   git Clone http://github.com/tgalal/Yowsup.git.

2) Set up Yowsup Configuration.
   ['Yowsup']
   CC = Country_code
   phone = phone_number
   password = Your_password.

Spdt

PAGE No.: _____

DATE : ____/____/____

Code

```
import RPi.GPIO as GPIO
from yowsup.layers.interface import YowInterfaceLayer,
    protocolEntityCallback.
from yowsup.layers.protocol_messages.protocolentities.
    import TextMessageProtocolEntity
from yowsup.layers import YowLayerEvent
from yowsup.layers.protocol_presence.protocolentities
    import PresenceProtocolEntity
# GPIO Setup
GPIO.setmode(GPIO.BCM)
LED_PIN = 18
GPIO.setup(LED_PIN, GPIO.OUT)
class WhatsAppBotLayer(YowInterfaceLayer):
@ProtocolEntityCallback("message")
    def onMessage(self,message):
        if isinstance(message, TextMessageProtocolEntity):
            incoming_msg = message.getBody().lower()
        if 'on' in incoming_msg: GPIO.output(LED_PIN)
            self.sendMessage(message.getFrom(), "LED is now ON")
        elif 'off' in incoming_msg: GPIO.output(LED_PIN)
            self.sendMessage(message.getFrom(), "Send 'on' or
                'off' to control the LED.")
        self.toLower(PresenceProtocolEntity())
        # send presence update.
```

Spdt

PAGE No.: _____

DATE : ____/____/____

```
def send_message(self, to, body):
    entity = TextMessageProtocolEntity(body, to).self.toLower(entity)
if __name__ == "__main__":
    from yowsup.stacks import YowStackBuilder
    from yowsup.layers.auth import AuthError
    from yowsup.layers.network import YowNetworkLayer
    from yowsup.common import YowConstants.
stack = YowStackBuilder().pushDefaultLayers(True).build()
stack.setCredentials(("yourphonenumber", "Password"))
bot_layer = WhatsAppBotLayer()
stack.broadcastEvent(YowLayerEvent(YowNetwork
    Layer.EVENT_STATE_CONNECT))

try:
    stack.loop
except AuthError as e:
    print("Auth Error: %e")
```

Conclusion

The WhatsApp-Controlled LED Project showcases how simple IoT Tasks can be managed remotely using messaging platforms.

Spdt PRACTICAL

PAGE No.: _____

DATE : __/__/__

Aim Fingerprint Sensor Interfacing with Raspberry Pi

Abstract This Demonstrates how to interface a fingerprint Sensor with a Raspberry Pi to capture and authenticate fingerprints. The System Allows for user identification by matching fingerprint data against a database, providing a secure and reliable authentication mechanism for various applications, such as access control or attendance systems. The fingerprint sensor is connected to the Raspberry Pi via GPIO pins and python is used for data processing and Sensor Communication.

Required Component:-

1) Raspberry Pi
2) Fingerprint Sensor
3) Adafruit fingerprint library
4) Serial Communication library
5) SQLite / MYSQL

Steps for Fingerprint Sensor Interfacing

1) Sensor wiring
   - Connect VCC, GND, TX and RX to GPIO 15 and 14 respectively

2) Installed libraries.

```
Sudo pip3 install pyserial
Sudo pip3 install.
```

PAGE No.: _____

Spdt

DATE : ____/____/____

Code:-

```
import time
import serial
import RPi.GPIO as GPIO
from adafruit_fingerprint import Adafruit_Fingerprint
#Setup Serial Communication with the Sensor.
uart = serial.Serial("/dev/ttyS0", baudrate=57600, timeout=1)
finger = Adafruit_Fingerprint(uart)
# GPIO Setup
GPIO.setmode(GPIO.BCM)
LED_PIN = 18
GPIO.setup(LED_PIN, GPIO.OUT)
def enroll_finger():
    """Enrolls a new fingerprint in the System"""
    print("Place finger on the Sensor...")
    while finger.get_image() != Adafruit_Fingerprint.OK:
        pass
    if finger.image_2_tz(1) != Adafruit_Fingerprint.OK:
        return False
    print("Remove finger...")
    time.sleep(2)
    print("Place the Same finger again....")
    while finger.get_image() != Adafruit_Fingerprint.OK:
        pass
    if finger.image_2_tz(2) != Adafruit_Fingerprint.OK:
        return False.
    if finger.create_model() != Adafruit_Fingerprint.OK:
        return False.
```

```
if finger.store_model(1) != Adafruit_Fingerprint.OK:
    return False
print("Fingerprint enrolled successfully!")
    return True

def verify_finger():
    """Verifies if the scanned fingerprint matches a stored one."""
    print("Place finger on sensor for verification...")
    if finger.get_image() == Adafruit_Fingerprint.OK:
        finger.image_2_tz(1)
        if finger.finger_search() == Adafruit_Fingerprint.OK:
            print("Fingerprint matched")
            GPIO.output(LED_PIN, GPIO.HIGH)
        else:
            print("No match found")
            GPIO.output(LED_PIN, GPIO.LOW)
    else:
        print("Error reading fingerprint")

if __name__ == "__main__":
    while True:
        print("Press 1 to Enroll finger, 2 to verify Finger:")
        Choice = input()
        if Choice == "1":
            enroll_finger()
        elif Choice == '2':
            verify_finger()
```

```
else:
    print("Invalid choice, please try again.")
```

Conclusion:-

The fingerprint Sensor project demonstrates Secure and reliable biometric authentication on the Raspberry Pi. Both projects highlights the versatility of Raspberry Pi in Creating practical, real-world applications through ~~mixing~~ minimal hardware and python-based Software integration.