Edge Computing Lab

Class: TY-AIEC

School of Computing, MIT Art Design Technology University

Academic Year: 2024-25

Experiment No. 1

Title: "Hello World" to Raspberry Pi

Raspbian OS Installation and Configuration on Raspberry Pi 4 and familiar with Raspberry Pi 4 GPIO's

Introduction

This manual provides a comprehensive guide for undergraduate students to understand the OS installation process using the Raspbian OS Imager, and configure remote access tools like PuTTY and VNC Viewer on Raspberry Pi 4.

Prerequisites

- Raspberry Pi 4
- MicroSD card (minimum 8 GB)
- Raspbian OS Imager software
- PuTTY for SSH access
- VNC Viewer for remote desktop access
- Internet connection
- Additional peripherals (monitor, keyboard, mouse, etc.)

Experiment Steps

Part 1: Installing Raspbian OS

Detailed steps on using the Raspbian OS Imager to select the correct version of the OS and write it to the MicroSD card. Illustrations will include screenshots of the imager interface and selection process.

Part 2: Setting Up Raspberry Pi

Instructions on inserting the MicroSD card into the Raspberry Pi, connecting all necessary peripherals, and completing the initial boot process. This section will include diagrams showing how to connect the Raspberry Pi to various peripherals.

Part 3: Configuring PuTTY for SSH Access

Step-by-step guide on installing PuTTY, finding the Raspberry Pi's IP address, and establishing an SSH connection. This section will contain figures illustrating the PuTTY configuration settings.

Part 4: Setting Up VNC Viewer

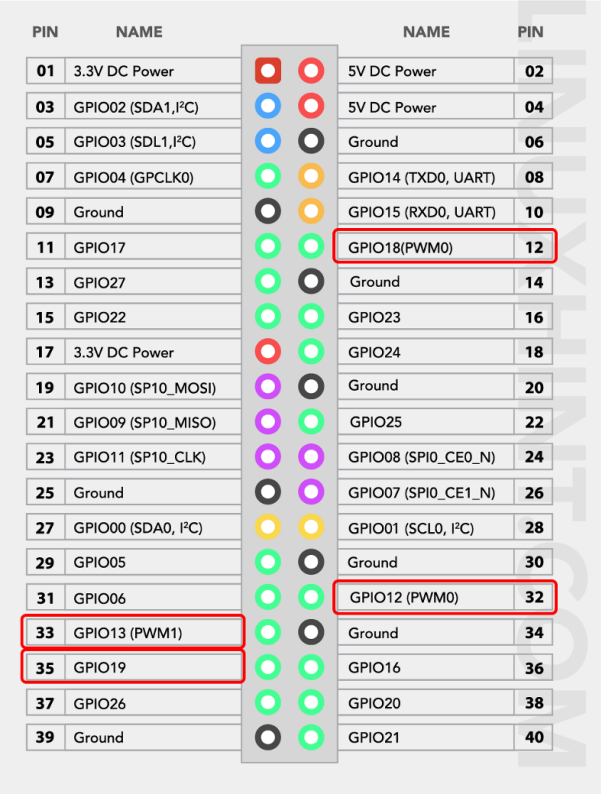
Detailed instructions on enabling VNC on the Raspberry Pi through the terminal or Raspberry Pi configuration settings, and connecting via VNC Viewer. Screenshots will guide the user through the VNC setup and connection process.

Part 5: Interface the LED with Raspberry Pi 4 on GPIO14 using BCM

Experiments tis section designed to introduce students to the fundamentals of hardware interfacing using the Raspberry Pi 4. By the end of this lab, students will learn how to control an external LED using the GPIO pins of the Raspberry Pi.

Theory

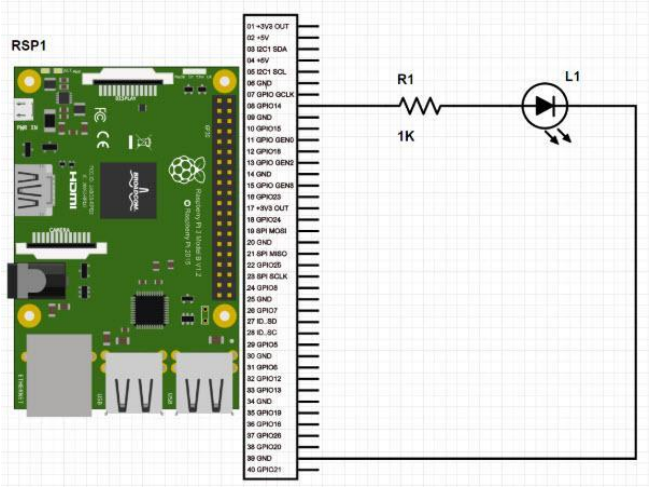
GPIO (General Purpose Input/Output) pins on the Raspberry Pi are used for interfacing with other electronic components. BCM numbering refers to the pin numbers in the Broadcom SOC channel, which is a more consistent way to refer to the GPIO pins across different versions of the Raspberry Pi.



PIN	NAME	NAME	PIN
01	3.3V DC Power	5V DC Power	02
03	GPIO02 (SDA1,I²C)	5V DC Power	04
05	GPIO03 (SDL1,I²C)	Ground	06
07	GPIO04 (GPCLK0)	GPIO14 (TXD0, UART)	08
09	Ground	GPIO15 (RXD0, UART)	10
11	GPIO17	GPIO18(PWM0)	12
13	GPIO27	Ground	14
15	GPIO22	GPIO23	16
17	3.3V DC Power	GPIO24	18
19	GPIO10 (SP10_MOSI)	Ground	20
21	GPIO09 (SP10_MISO)	GPIO25	22
23	GPIO11 (SP10_CLK)	GPIO08 (SPI0_CE0_N)	24
25	Ground	GPIO07 (SPI0_CE1_N)	26
27	GPIO00 (SDA0, I²C)	GPIO01 (SCL0, I²C)	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12 (PWM0)	32
33	GPIO13 (PWM1)	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Circuit Diagram

An illustrative diagram showing how to connect the LED to the Raspberry Pi GPIO14 pin through a resistor using a breadboard will be included here.



Experiment Procedure

Detailed steps for setting up the Raspberry Pi, connecting the LED to the GPIO14 pin through the resistor, and securing all connections on the breadboard.

Python Code

A simple Python script to control the LED by turning it on and off will be provided, demonstrating the use of GPIO library and BCM pin numbering.

```
import RPi.GPIO as GPIO
import time

# Use BCM GPIO references instead of physical pin numbers
GPIO.setmode(GPIO.BCM)

# Define GPIO signal to use (Physical pin 7 corresponds to BCM GPIO 4)
GPIO_LED = 4

# Set up the GPIO channel as output
GPIO.setup(GPIO_LED, GPIO.OUT)

try:
    # Loop to blink the LED on and off
    while True:
        # Turn LED on
        GPIO.output(GPIO_LED, True)
        print("LED ON")
        time.sleep(1)  # Sleep for 1 second

        # Turn LED off
        GPIO.output(GPIO_LED, False)
        print("LED OFF")
        time.sleep(1)  # Sleep for 1 second

except KeyboardInterrupt:
    # Clean up on Ctrl+C exit
    GPIO.cleanup()
```

Python Code to display 'Hello World' on LCD

```
import smbus

import time


# Define the I2C address of the LCD

LCD_ADDRESS = 0x3E

RGB_ADDRESS = 0x62


# Initialize the I2C bus

bus = smbus.SMBus(1)


def send_command(cmd):

      """Send a command to the LCD."""

      bus.write_byte_data(LCD_ADDRESS, 0x00, cmd)


def send_data(data):

      """Send data to the LCD."""

      bus.write_byte_data(LCD_ADDRESS, 0x40, data)


def set_rgb(r, g, b):

      """Set the backlight color of the LCD."""

      bus.write_byte_data(RGB_ADDRESS, 0x00, 0x00)

      bus.write_byte_data(RGB_ADDRESS, 0x01, 0x00)

      bus.write_byte_data(RGB_ADDRESS, 0x08, 0xAA)

      bus.write_byte_data(RGB_ADDRESS, 0x04, r)
```

```
    bus.write_byte_data(RGB_ADDRESS, 0x03, g)
    bus.write_byte_data(RGB_ADDRESS, 0x02, b)


def initialize_lcd():
    """Initialize the LCD."""
    send_command(0x01)  # Clear display
    time.sleep(0.05)
    send_command(0x38)  # Function set: 8-bit, 2 lines
    send_command(0x0C)  # Display ON, Cursor OFF, Blink OFF
    send_command(0x06)  # Entry mode set: increment, no shift
    time.sleep(0.05)


def write_message(line1, line2):
    """Write a message to the LCD."""
    send_command(0x80)  # Move cursor to the beginning of the first
line
    for char in line1:
    send_data(ord(char))  # Send each character as data to the LCD


    send_command(0xC0)  # Move cursor to the beginning of the second
line
    for char in line2:
    send_data(ord(char))  # Send each character as data to the LCD


if __name__ == "__main__":
    try:
```

```
initialize_lcd()

set_rgb(0, 255, 255)  # Set backlight to greenish color

write_message("Hello", "World")  # Display message

while True:

      time.sleep(1)

except KeyboardInterrupt:

send_command(0x01)  # Clear display

set_rgb(0, 0, 0)  # Turn off backlight

print("Exiting...")
```

Verification and Testing

Steps to verify the successful installation of the Raspbian OS, and the functionality of SSH and VNC connections, including sample commands and expected outputs.

Troubleshooting Tips

Common issues and their solutions related to OS installation, SSH, and VNC connections, supported by troubleshooting flowcharts and diagrams.

Observation

Students are expected to observe the LED turning on and off in response to the Python script, demonstrating the successful interfacing and control of external hardware with the Raspberry Pi

Conclusion

A summary of the key learning points from the manual and encouragement for students to explore further applications and configurations of the Raspberry Pi 4.

WriteUp

MIT SCHOOL OF COMPUTING
Rajbaug, Loni-Kalbhor, Pune

MIT-ADT UNIVERSITY
PUNE, INDIA
A Leap Towards World Class Education

Name : Sudip Satish kande Class : TY (AIEC)
Roll No : 2223118 Sub : ECL

Experiment No 1

Introduction to Rasberry Pi, OS installation, Putty & VNC

① What are the key hardware component of a Raspberry Pi & how does it differ from a traditional computer.

Processor - An Arm based broadcom SOC (System on chip)
RAM - Varies by model
Storage - Uses a micro SD card
GPIO Pin - 40 pins GPIO
USB Port - Used for periferal like keyboard, mouse or storage device
HDMI Port - For video output
Power Input - Usually via USB-C
Networking - Include ethernet & wifi

② Difference from traditional computer

Size - Rasberry Pi is compact.
Storage - Uses a SD card instead of built-in-storage
Processing Power - less powerful than a typical PC.

Expandability - No PCIe slots, instead it has GPIO for hardware project.
low power consumption - Runs on 5V whereas traditional PCs require much higher power

2) Describe the step by step process for installing an OS on a Raspberry Pi using Raspberry Pi imager.

(i) Download & install Raspberry Pi imager.
(ii) Insert a microSD card into your PC
(iii) Open imager, select OS & storage then click write
(iv) Insert the microSD card into the Raspberry Pi & boot it.

3) How can Putty be used to establish an SSH connection with Raspberry Pi? what are the default login credential

1) Enable SSH (Create a blank SSH file in the boot partition)
2) Find ip address (host name - I)
3) Open Putty, enter, Raisberry Pi IP, set port 22 & connect.
4) login - Username - Pi, Password - Raspberry.

4 Explain the Purpose of VNC in Rasberry Pi How can you enable & access the Raspberry Pi remotely using VNC.

i) Enable VNC: sudo raspi config > Interfacing > Options > Enable VNC.
ii) Install VNC viewer on your PC.
iii) Find Ip address (hostname -I)
iv) Open VNC viewer, enter Pi-Ip:1 and login

5 What are the advantage of using headless mode for Rasp Pi, & how can it be configured

Headless mode configuration Advantage -

No monitor needed
Remote access via SSH/VNC
Save Space, ideal for IoT
Setup ->

1. Flash OS to microSD
2. Create SSH file in boot partition.
3. Create WiFi config
4. Boot Pi, find IP & Connect via SSH (ssh pi@pi_Ip).