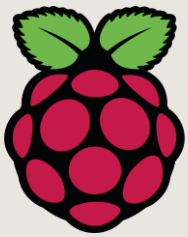
SMEC
Embedded Systems

INTRODUCTION TO RASPBERRY PI

[TEXT BOOK]



https://smeclabs.ac.in/

TABLE OF CONTENTS

Chapter 1: Introduction to Raspberry Pi

What is a Raspberry Pi?

A Raspberry Pi is a small, affordable, single-board computer developed by the Raspberry Pi Foundation. It was originally designed to promote the teaching of basic computer science in schools and developing countries, but its versatility, low cost, and wide range of applications have made it popular among hobbyists, educators, and professionals.

Key Features of Raspberry Pi:

- Compact Size : Roughly the size of a credit card, making it highly portable and easy to integrate into various projects.
- Cost-Effective :Typically priced between $5 and $75, depending on the model and features.
- Versatility: Capable of functioning as a desktop computer, media centre, server, or for various DIY electronics projects.
- GPIO Pins: General Purpose Input/output pins allow for interfacing with sensors, motors, LEDs, and other electronics, enabling the creation of complex projects.
- Connectivity : Equipped with USB ports, HDMI output, Ethernet, Wi-Fi, Bluetooth, and other interfaces for connectivity and peripheral support.
- Software : Runs on Linux-based operating systems, primarily Raspberry Pi OS (formerly Raspbian), and supports other OSes like Ubuntu, Windows IoT, and more.
- Educational Focus: Aimed at providing an accessible platform for learning programming and computer science.

Applications of Raspberry Pi:

- Education: Teaching programming, computer science, and electronics.
- Home Automation: Controlling lights, appliances, and security systems.
- Media Centers: Setting up home entertainment systems using software like Kodi.
- Gaming: Retro gaming consoles using emulators.
- IoT Projects: Building Internet of Things devices for smart homes and industrial applications.
- Prototyping: Rapid development and testing of hardware and software projects.

The Raspberry Pi's flexibility, affordability, and wide community support make it an ideal choice for a broad range of applications, from simple educational tools to complex industrial systems.

The Raspberry Pi is a ground breaking series of small single-board computers developed by the Raspberry Pi Foundation, a UK-based charity. Here is a detailed timeline of its history and evolution:

Origins and Concept (2006-2011)

- **2006**: The idea for the Raspberry Pi was conceived by Eben Upton, Rob Mullins, Jack Lang, and Alan Mycroft at the University of Cambridge's Computer Laboratory. They aimed to create a low-cost device to improve computer science education.

- **2008**: Development of the first prototype began. The team experimented with different designs to achieve a balance between affordability and functionality.

Launch and Early Models (2012-2014)

- **2012**: The first model, Raspberry Pi Model B, was launched on February 29, 2012. It featured:
  - ✓ Broadcom BCM2835 SoC with a 700 MHz ARM1176JZF-S CPU
  - ✓ 256 MB of RAM
  - ✓ Two USB ports and an Ethernet port
  - ✓ Composite video and HDMI output

- **2013**: The Raspberry Pi Model A was released, offering a cheaper, more compact version with:
  - ✓ No Ethernet port
  - ✓ One USB port
  - ✓ 256 MB of RAM

- **2014**: The Raspberry Pi Model B+ was introduced with improvements:
  - ✓ Four USB ports
  - ✓ 512 MB of RAM
  - ✓ MicroSD slot replacing the full-size SD slot
  - ✓ Improved power consumption and audio

Significant Advancements (2015-2016)

•**2015**: The Raspberry Pi 2 Model B was launched with significant upgrades:
- ✓ Broadcom BCM2836 SoC with a 900 MHz quad-core ARM Cortex-A7 CPU
- ✓ 1 GB of RAM
- ✓ Backward compatibility with previous models

•**2016**: The Raspberry Pi 3 Model B was introduced with:
- ✓ Broadcom BCM2837 SoC with a 1.2 GHz 64-bit quad-core ARM Cortex-A53 CPU
- ✓ Integrated Wi-Fi and Bluetooth

Recent Models and Innovations (2017-2021)

•**2017**: The Raspberry Pi Zero W was launched, adding Wi-Fi and Bluetooth to the Pi Zero.

•**2018**: The Raspberry Pi 3 Model B+ was released, offering:
- ✓ Improved 1.4 GHz 64-bit quad-core ARM Cortex-A53 CPU
- ✓ Faster Ethernet and dual-band Wi-Fi

•**2019**: The Raspberry Pi 4 Model B was a major upgrade:
- ✓ Broadcom BCM2711 SoC with a 1.5 GHz 64-bit quad-core ARM Cortex-A72 CPU
- ✓ Multiple RAM options (2 GB, 4 GB, 8 GB)
- ✓ USB 3.0 ports, Gigabit Ethernet, and dual micro-HDMI ports

•**2020**: The Raspberry Pi 400 was launched, integrating the computer into a compact keyboard:
- ✓ Similar specs to the Raspberry Pi 4 but with a 1.8 GHz CPU

•**2021**: The Raspberry Pi Pico, a microcontroller board, was introduced:
- ✓ RP2040 microcontroller chip designed by Raspberry Pi
- ✓ Dual-core ARM Cortex-M0+ processor
- ✓ GPIO pins for hardware projects

Latest Developments (2022-2024)

•**2022**: The Raspberry Pi Zero 2 W was released:

- Broadcom BCM2710A1 SoC with a 1 GHz quad-core ARM Cortex-A53 CPU
- 512 MB of RAM
- Integrated Wi-Fi and Bluetooth

•**2023**: Raspberry Pi introduced the Compute Module 4 (CM4) with:

- Multiple configurations for RAM and storage
- Enhanced connectivity options
- Broadcom BCM2711 SoC with a 1.5 GHz 64-bit quad-core ARM Cortex-A72 CPU
- Focused on industrial applications and custom designs

•**2024**: The Raspberry Pi 5 Model B is announced with substantial performance improvements:

- Broadcom BCM2838 SoC with a 2.0 GHz 64-bit quad-core ARM Cortex-A76 CPU
- Up to 16 GB of RAM options
- Enhanced GPU capabilities for better graphics performance
- USB 4.0 ports and faster networking options

Impact and Legacy

•**Educational Tool**: The Raspberry Pi has been widely adopted in education, promoting programming and STEM skills.

•**Community and Ecosystem**: A vibrant community has grown around the Raspberry Pi, contributing to software, projects, and accessories.

•**Commercial Applications**: Beyond education and hobbies, the Raspberry Pi is used in industrial automation, IoT, and prototyping.

Raspberry Pi Models and Their Differences

The Raspberry Pi series has undergone several iterations since its inception, with each model offering different features and improvements. Here's an overview of the various Raspberry Pi models and their key differences:

Raspberry Pi Model B (2012)

- **SoC**: Broadcom BCM2835
- **CPU**: 700 MHz ARM1176JZF-S
- **RAM**: 256 MB (later upgraded to 512 MB)
- **Ports**: 2 USB ports, 1 Ethernet port
- **Storage**: SD card slot
- **Connectivity**: None

Raspberry Pi Model B+ (2014)

- **SoC**: Broadcom BCM2835
- **CPU**: 700 MHz ARM1176JZF-S
- **RAM**: 512 MB
- **Ports**: 4 USB ports, 1 Ethernet port
- **Storage**: MicroSD card slot
- **Connectivity**: None

Raspberry Pi Zero (2015)

- **SoC**: Broadcom BCM2835
- **CPU**: 1 GHz ARM11
- **RAM**: 512 MB
- **Ports**: Mini HDMI, USB On-The-Go
- **Storage**: MicroSD card slot
- **Connectivity**: None

Raspberry Pi 3 Model B+ (2018)

- **SoC**: Broadcom BCM2837B0
- **CPU**: 1.4 GHz 64-bit quad-core ARM Cortex-A53
- **RAM**: 1 GB
- **Ports**: 4 USB ports, 1 Gigabit Ethernet port (limited to 300 Mbps)
- **Storage**: MicroSD card slot
- **Connectivity**: Dual-band Wi-Fi 802.11ac, Bluetooth 4.2

Raspberry Pi Model A (2013)

- **SoC**: Broadcom BCM2835
- **CPU**: 700 MHz ARM1176JZF-S
- **RAM**: 256 MB
- **Ports**: 1 USB port
- **Storage**: SD card slot
- **Connectivity**: None

Raspberry Pi 2 Model B (2015)

- **SoC**: Broadcom BCM2836
- **CPU**: 900 MHz quad-core ARM Cortex-A7
- **RAM**: 1 GB
- **Ports**: 4 USB ports, 1 Ethernet port
- **Storage**: MicroSD card slot
- **Connectivity**: None

Raspberry Pi 3 Model B (2016)

- **SoC**: Broadcom BCM2837
- **CPU**: 1.2 GHz 64-bit quad-core ARM Cortex-A53
- **RAM**: 1 GB
- **Ports**: 4 USB ports, 1 Ethernet port
- **Storage**: MicroSD card slot
- **Connectivity**: Wi-Fi 802.11n, Bluetooth 4.1

Raspberry Pi 4 Model B (2019)

- **SoC**: Broadcom BCM2711
- **CPU**: 1.5 GHz 64-bit quad-core ARM Cortex-A72
- **RAM**: 2 GB, 4 GB, or 8 GB
- **Ports**: 2 USB 3.0 ports, 2 USB 2.0 ports, 1 Gigabit Ethernet port
- **Storage**: MicroSD card slot
- **Connectivity**: Dual-band Wi-Fi 802.11ac, Bluetooth 5.0
- **Video Output**: 2 micro-HDMI ports supporting 4K resolution

Raspberry Pi 400 (2020)

- **SoC**: Broadcom BCM2711
- **CPU**: 1.8 GHz 64-bit quad-core ARM Cortex-A72
- **RAM**: 4 GB
- **Ports**: 3 USB 3.0 ports, 1 USB 2.0 port, 1 Gigabit Ethernet port
- **Storage**: MicroSD card slot
- **Connectivity**: Dual-band Wi-Fi 802.11ac, Bluetooth 5.0
- **Form Factor**: Integrated into a compact keyboard

Raspberry Pi Pico (2021)

- **Microcontroller**: RP2040
- **CPU**: Dual-core ARM Cortex-M0+
- **RAM**: 264 KB
- **Storage**: 2 MB onboard flash memory
- **Ports**: 26 multi-function GPIO pins
- **Connectivity**: None (no built-in Wi-Fi or Bluetooth)

Raspberry Pi Zero 2 W (2022)

- **SoC**: Broadcom BCM2710A1
- **CPU**: 1 GHz quad-core ARM Cortex-A53
- **RAM**: 512 MB
- **Ports**: Mini HDMI, USB On-The-Go
- **Storage**: MicroSD card slot
- **Connectivity**: Wi-Fi 802.11n, Bluetooth 4.2

Raspberry Pi 5 Model B (2024)

- **SoC**: Broadcom BCM2838
- **CPU**: 2.0 GHz 64-bit quad-core ARM Cortex-A76
- **RAM**: 4 GB, 8 GB, or 16 GB
- **Ports**: 2 USB 4.0 ports, 2 USB 3.0 ports, 1 Gigabit Ethernet port
- **Storage**: MicroSD card slot
- **Connectivity**: Dual-band Wi-Fi 802.11ax (Wi-Fi 6), Bluetooth 5.2
- **Video Output**: Dual micro-HDMI ports supporting 4K resolution at 60 Hz

Key Differences Across Models

1.Processing Power: Each successive model generally features a more powerful CPU, with improvements in speed and efficiency.

2.RAM: Later models offer more RAM, with options up to 16 GB in the Raspberry Pi 5 Model B.

3.Connectivity: Integrated Wi-Fi and Bluetooth were introduced in the Raspberry Pi 3 Model B and improved in later models.

4.Ports: The number and type of USB ports have evolved, with newer models offering faster USB 3.0 and USB 4.0 ports.

5.Storage: Transition from full-size SD to microSD for storage across models.

6.Video Output: Introduction of dual HDMI outputs in the Raspberry Pi 4 Model B for enhanced display capabilities.

7.Form Factor: Variations include the compact Raspberry Pi Zero, the keyboard-integrated Raspberry Pi 400, and the microcontroller Raspberry Pi Pico.

Applications and Uses of Raspberry Pi

The versatility, affordability, and compact size of the Raspberry Pi make it an ideal platform for a wide range of applications. Here are some of the most popular and innovative uses:

1. Education

- **Programming and Coding**: Ideal for teaching programming languages like Python, Scratch, and JavaScript.
- **STEM Projects**: Widely used in schools for science, technology, engineering, and math projects.
- **Computer Science**: Enables students to learn about operating systems, networking, and hardware.

2. Home Automation

- **Smart Home Systems**: Controls lighting, heating, and security systems.
- **Voice Assistants**: Integrates with voice assistants like Alexa or Google Assistant for voice-controlled home automation.
- **Surveillance**: Used for setting up security cameras and monitoring systems.

3. Media Centers

- **Kodi Media Center**: Runs Kodi to turn the Raspberry Pi into a powerful media center.
- **Streaming**: Streams movies, music, and TV shows from various services.
- **Retro Gaming Consoles**: Emulates classic gaming consoles using software like RetroPie.

4. Internet of Things (IoT)

- **Smart Sensors**: Interfaces with various sensors for temperature, humidity, motion, etc.
- **IoT Gateways**: Acts as a gateway for IoT devices, collecting and processing data.
- **Remote Monitoring**: Monitors environmental conditions, industrial processes, and more.

5. Robotics

- **Control Systems**: Acts as the brain for robots, controlling motors, sensors, and other components.
- **Autonomous Vehicles**: Used in DIY drones, rovers, and other autonomous projects.
- **Educational Robots**: Powers educational robotics kits for teaching robotics and engineering.

6. Networking

- **Network Attached Storage (NAS)**: Serves as a low-cost NAS solution for file storage and sharing.
- **VPN Server**: Configured as a VPN server to secure internet connections.
- **Ad Blocker**: Runs Pi-hole to block ads and trackers at the network level.

7. Industrial Applications

- **Prototyping**: Used for rapid prototyping of industrial control systems and IoT devices.
- **Edge Computing**: Processes data at the edge of the network, reducing latency and bandwidth usage.
- **Automation**: Controls and monitors industrial processes, machinery, and equipment.

8. Personal Projects and Hobbies

- **Weather Stations**: Gathers and displays weather data from various sensors.
- **Home Labs**: Used in personal lab setups for experimenting with networking, servers, and more.
- **Art and Creativity**: Powers interactive art installations, music projects, and other creative endeavours.

9. Web Servers

- **LAMP Stack**: Sets up a LAMP (Linux, Apache, MySQL, PHP) server for web hosting.
- **Development Server**: Acts as a local server for web development and testing.
- **WordPress Hosting**: Hosts WordPress sites for personal or small business use.

10. Learning and Development

- **Linux**: Provides a hands-on platform for learning Linux and its various distributions.
- **Networking**: Teaches networking concepts and practical skills through projects and experiments.
- **Cybersecurity**: Used in cybersecurity training for practicing ethical hacking and network defence.

The Raspberry Pi's wide range of applications highlights its versatility and capability as a tool for education, hobby projects, professional development, and industrial use. Its low cost and extensive community support make it accessible for anyone looking to explore computing, programming, and electronics.

Unboxing and Understanding the Components

When you receive your Raspberry Pi, unboxing and familiarizing yourself with the components is the first step. Here's a detailed guide to help you understand each component and its purpose:

What's Inside the Box?

Depending on the model and the bundle you purchased, your Raspberry Pi kit may include the following components:

1.Raspberry Pi Board
2.MicroSD Card
3.Power Supply
4.Case
5.HDMI Cable
6.Heat Sinks
7.GPIO Pins and Accessories
8.User Manual/Quick Start Guide

Understanding Each Component

1.Raspberry Pi Board

- **CPU/GPU**: The central processing unit (CPU) and graphics processing unit (GPU) are integrated into the system-on-chip (SoC). This is the brain of the Raspberry Pi.
- **RAM**: Memory that helps in multitasking and running applications. The amount of RAM varies with different models.
- **USB Ports**: Used to connect peripherals such as a keyboard, mouse, USB drives, and other devices.
- **Ethernet Port**: For wired network connections (available in models like the B series).
- **HDMI Ports**: Connects the Raspberry Pi to a monitor or TV. Models like the Raspberry Pi 4 have dual micro-HDMI ports.
- **Power Supply Port**: Typically a micro-USB or USB-C port to power the Raspberry Pi.

- **Audio Jack**: Outputs audio to speakers or headphones (available in most models).
- **GPIO Pins**: General Purpose Input/output pins for connecting various hardware components and sensors.
- **Display Connector (DSI)**: Interface for connecting a Raspberry Pi touchscreen display.
- **Camera Connector (CSI)**: Interface for connecting the Raspberry Pi Camera Module.
- **MicroSD Card Slot**: Storage for the operating system and files. The primary storage for the Raspberry Pi.

2. MicroSD Card

- ❑ **Function**: Acts as the primary storage device, holding the operating system and other necessary files.
- ❑ **Preloaded Software**: Some kits come with a microSD card preloaded with NOOBS (New Out Of the Box Software) to help you install the operating system.

3. Power Supply

- ❑ **Specifications**: The power supply provides the necessary power for the Raspberry Pi to function. Ensure it matches the required voltage and current specifications (e.g., 5V, 2.5A for Raspberry Pi 3, 5V, 3A for Raspberry Pi 4).
- ❑ **Connector Type**: Usually a micro-USB or USB-C connector depending on the Raspberry Pi model.

4. Case

- ❑ **Protection**: A case helps protect the Raspberry Pi board from physical damage and dust.
- ❑ **Accessibility**: Provides access to all ports and connectors while keeping the board secure.

5. HDMI Cable

- ❑ **Purpose**: Connects the Raspberry Pi to a monitor or TV for video output.

- ❑ **Type**: Depending on the model, it could be a full-sized HDMI, mini-HDMI, or micro-HDMI cable.

6. Heat Sinks

•**Cooling**: Heat sinks are used to dissipate heat from the CPU and other components, especially if you plan to overclock the board or use it for intensive tasks.

7. GPIO Pins and Accessories

•**Functionality**: The GPIO pins allow for the connection of various external components like sensors, LEDs, motors, and other electronics for hardware projects.

8. User Manual/Quick Start Guide

•**Guidance**: Provides essential information on setting up your Raspberry Pi, including initial boot instructions, safety information, and basic troubleshooting tips.

Unboxing Checklist

1.Check all components: Ensure all components listed in the kit are present.
2.Inspect the board: Look for any visible damage or manufacturing defects.
3.Read the manual: Familiarize yourself with the quick start guide or manual included in the kit.

Setting Up Your Raspberry Pi

1.Insert the MicroSD Card: Load the microSD card with the operating system into the card slot on the Raspberry Pi.
2.Connect the Monitor: Use the HDMI cable to connect your Raspberry Pi to a monitor or TV.
3.Attach Peripherals: Connect your keyboard, mouse, and any other peripherals to the USB ports.
4.Power Up: Plug in the power supply to the Raspberry Pi and connect it to a power source.
5.Initial Boot: The Raspberry Pi should start booting up, displaying the initial setup screen on the monitor.

Understanding the components of your Raspberry Pi is crucial for a smooth setup and effective usage. Each component has a specific role, and knowing how to connect and configure them will help you get the most out of your Raspberry Pi.

Installing the Operating System (Raspberry Pi OS)

Installing the operating system (OS) is a critical step to get your Raspberry Pi up and running. This guide will help you install the Raspberry Pi OS, the official operating system for Raspberry Pi.

What You'll Need:

- **Raspberry Pi** board
- **MicroSD card** (recommended at least 8 GB)
- **MicroSD card reader**
- **Computer** with internet access
- **Power supply** for the Raspberry Pi
- **HDMI cable** and monitor
- **Keyboard and mouse**

Steps to Install Raspberry Pi OS:

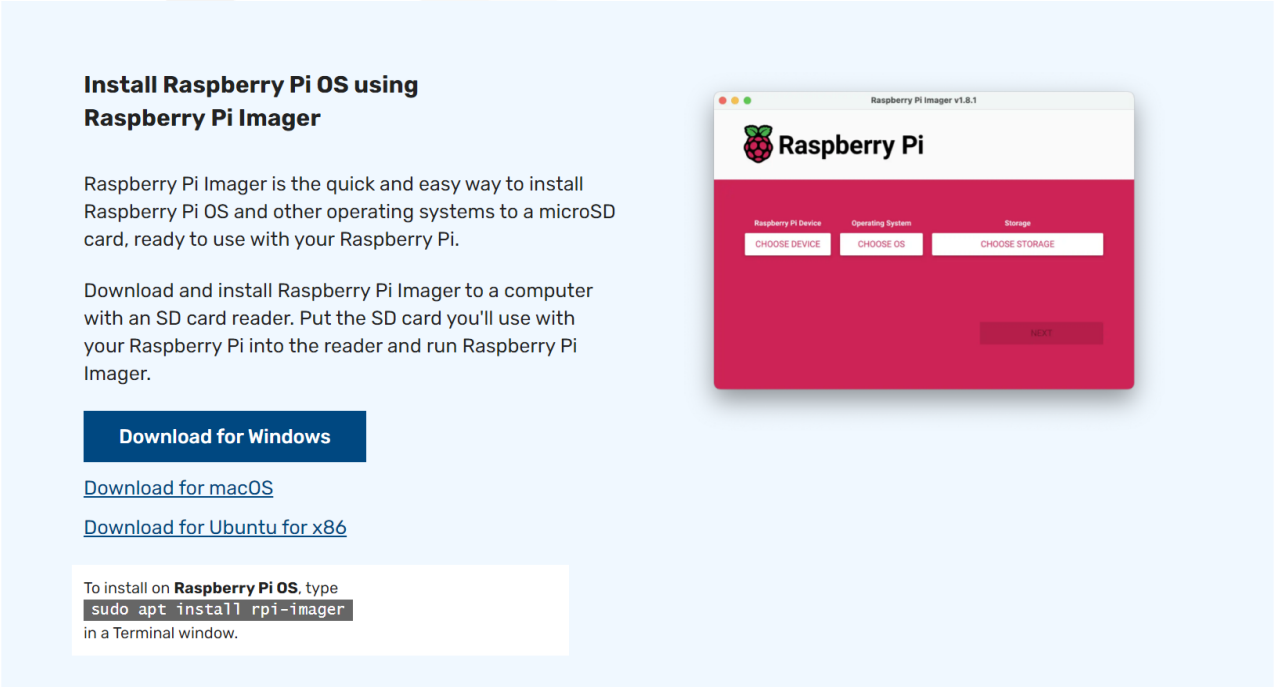
Prepare the MicroSD Card

1.Download the Raspberry Pi Imager:

- Visit the official Raspberry Pi website and download the Raspberry Pi Imager for your operating system (Windows, macOS, or Linux).

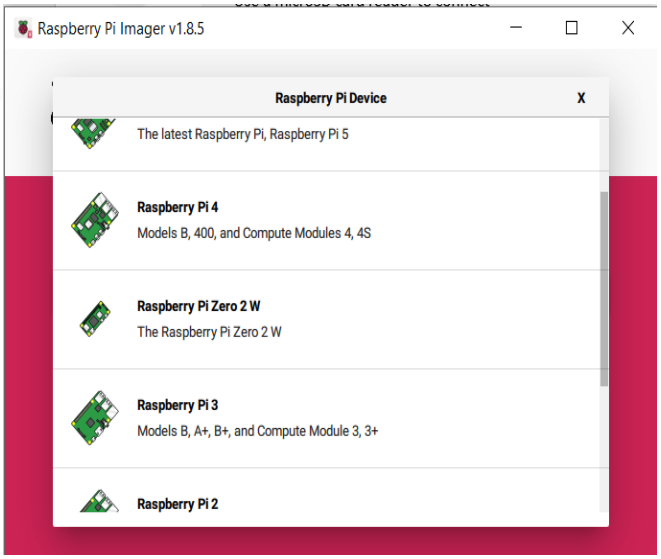
https://www.raspberrypi.com/software/

2. **Download and Install the Raspberry Pi Imager:**
   - Follow the installation instructions for your operating system to install the Raspberry Pi Imager.

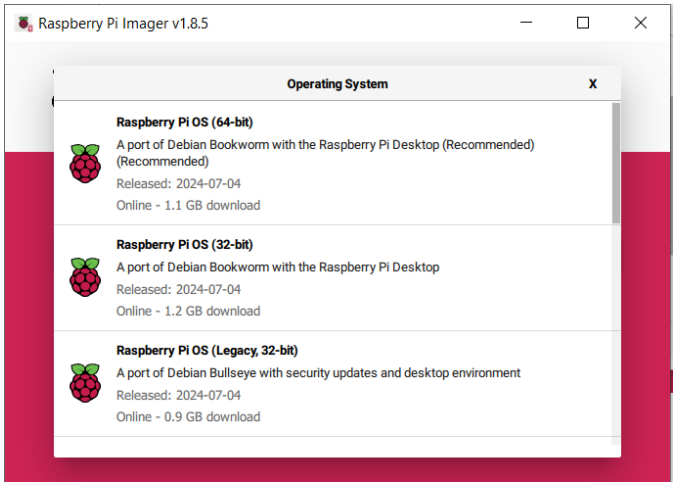


3. **Insert the MicroSD Card:**
   - Use a microSD card reader to connect the microSD card to your computer.

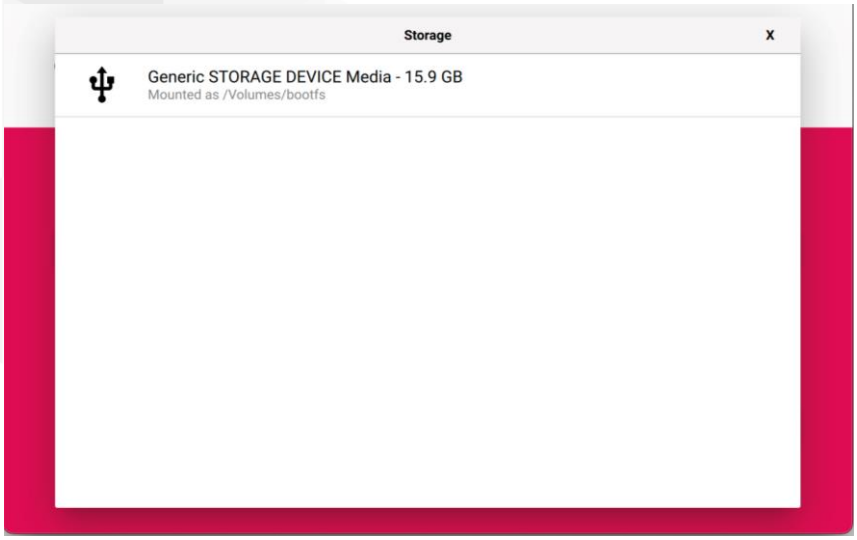
4. **Choose device:**
   - Click on "CHOOSE DEVICE" and select your Raspberry board model from the list.



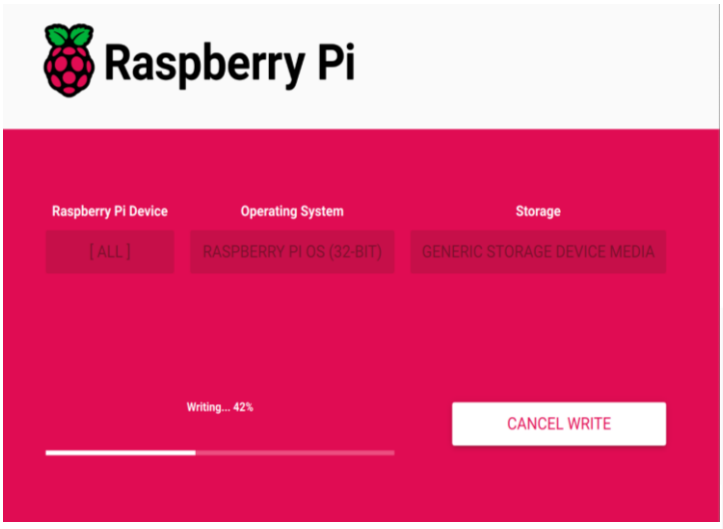
5. **Select Operating System:**
   - Click on "CHOOSE OS" and select "Raspberry Pi OS (32-bit)" or "Raspberry Pi OS (64-bit)" depending on your preference and your Raspberry Pi model. The standard "Raspberry Pi OS with desktop" is recommended for most users.



6. **Select Storage:**
   - Click on "CHOOSE SD CARD" and select your microSD card from the list.



7. **Write the OS:**
   - Click "WRITE" to start writing the Raspberry Pi OS to the microSD card. This process will format the card, so ensure any important data is backed up.



8. **Wait for the Process to Complete:**

The Imager will download the OS, write it to the microSD card, and verify the installation. This may take a few minutes.

1.Insert the MicroSD Card into the Raspberry Pi:

- Once the OS has been successfully written, remove the microSD card from the reader and insert it into the microSD card slot on the Raspberry Pi.

2.Connect Peripherals:

- Connect your monitor to the Raspberry Pi using the HDMI cable.
- Plug in your USB keyboard and mouse.

3.Power Up:

- Connect the power supply to the Raspberry Pi and plug it into a power source. The Raspberry Pi will start booting automatically.

4.Initial Setup:

- The Raspberry Pi OS will boot up and display the setup wizard. Follow the on-screen instructions to:
  1. Select your country, language, and time zone.
  2. Create a user account with a username and password.
  3. Set up Wi-Fi (if applicable).
  4. Update the software to ensure you have the latest features and security updates.

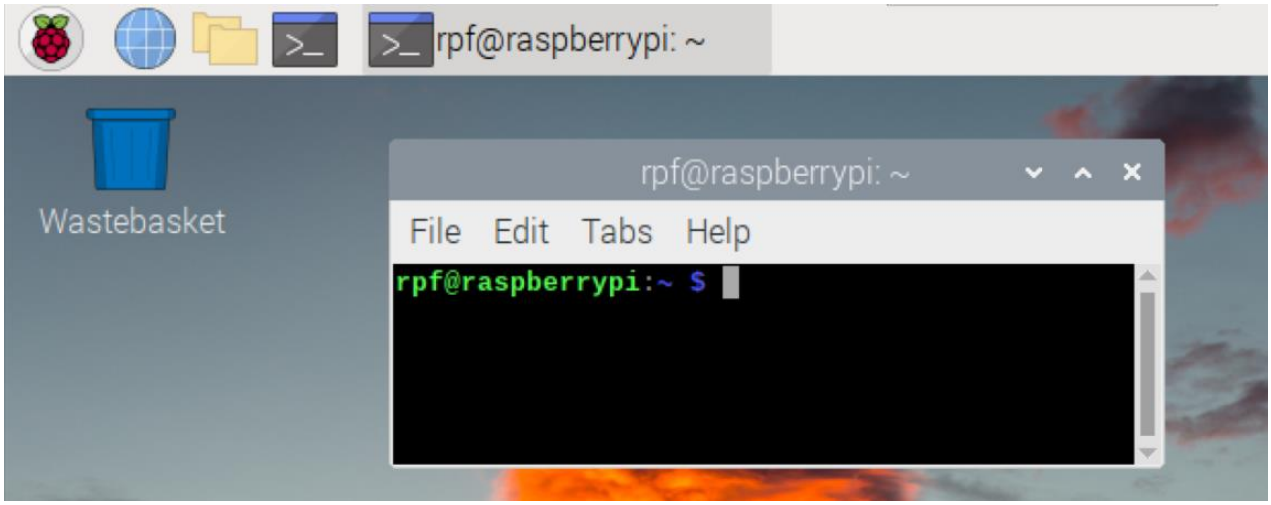
5.Enable Additional Features (Optional):

- You can enable additional features like VNC (for remote desktop access), SSH (for secure remote command line access), and configure the GPIO settings.

The Raspberry Pi OS is a Linux-based operating system, so understanding basic Linux commands is crucial for effectively managing and using your Raspberry Pi. This chapter introduces fundamental Linux commands that will help you navigate the file system, manage files and directories, and perform essential administrative tasks.

The Terminal emulator tool is used in this chapter. We can get this from the navigation tab. The terminal is a really useful application: it allows you to navigate file directories and control your Raspberry Pi using typed commands instead of clicking on menu options.

•To open a terminal window, click on the Terminal icon at the top of the screen, or select Accessories and then Terminal in the menu. You should see the following prompt, although the exact prompt will show the username you have chosen.



- **Navigating the File System**

a) pwd (Print Working Directory)

•**Usage**: pwd

•**Description**: Displays the current directory you are in.

b) ls (List)

•**Usage**: ls [options] [directory]
•**Description**: Lists the files and directories in the specified directory. If no directory is specified, it lists the contents of the current directory.
•**Options**:
- •-l: Long format listing
- •-a: Includes hidden files
- •-h: Human-readable file sizes

c) mkdir (make directory)

•**Usage**: mkdir [folder name]
•**Description**: TO create a folder in current directory .

d) cd (Change Directory)

•**Usage**: cd [directory]
•**Description**: Changes the current directory to the specified directory.
•**cd .. –** Go to previous directory

e) touch (Change Directory)

- **Usage**: touch [filename]
- **Description**: Creates an empty file with the specified name or updates the timestamp of an existing file.

```
pi@raspberrypi:~/NewFolder $ touch NewFile.txt
pi@raspberrypi:~/NewFolder $ ls
NewFile.txt
```

f) cp (Copy)

- **Usage**: cp [options] [source] [destination]
- **Description**: Copies files or directories from the source to the destination.
- **Options**:
- -r: Recursively copies directories and their contents

```
pi@raspberrypi:~/NewFolder $ cp NewFile.txt OtherFile.txt
pi@raspberrypi:~/NewFolder $ ls
NewFile.txt  OtherFile.txt
```

g) rm (Remove)

- **Usage**: rm [options] [file]
- **Description**: Removes the specified file.
- **Options**:
  - -r: Recursively removes directories and their contents
  - -f: Forces removal without prompting for confirmation

```
pi@raspberrypi:~/NewFolder $ rm NewFile.txt
pi@raspberrypi:~/NewFolder $ ls
OtherFile.txt
```

h) mv (Move/Rename)

- **Usage**: mv [source] [destination]
- **Description**: Moves or renames files or directories.

```
pi@raspberrypi:~/NewFolder $ mv OtherFile.txt /home/pi/
pi@raspberrypi:~/NewFolder $ cd ..
pi@raspberrypi:~ $ ls
NewFolder  OtherFile.txt
pi@raspberrypi:~ $
```

Updating Software

Keeping your Raspberry Pi's software up to date is crucial for security, stability, and accessing the latest features. This section will guide you through updating your Raspberry Pi OS and installed packages using command-line tools.

Update Package Lists

✓ **Update Package Lists**

- **Command**: sudo apt update
- **Description**: Fetches the latest package lists from the repositories, ensuring you have the most recent information about available packages and updates.

Upgrade Installed Packages

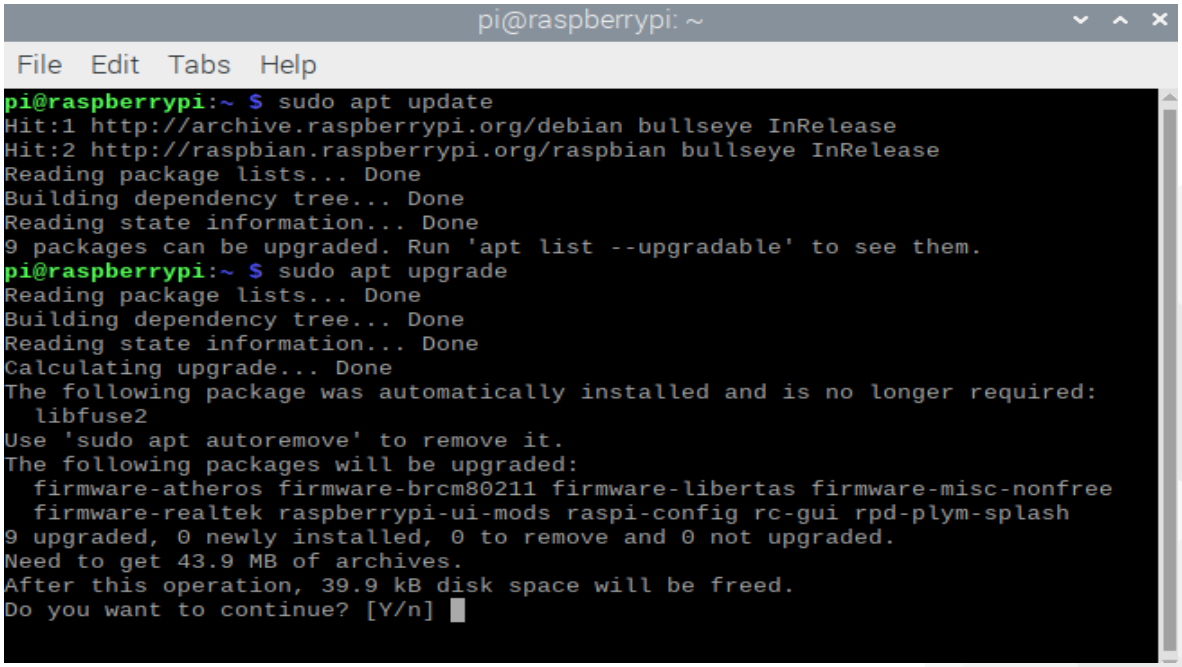
✓ **Upgrade Packages**

- **Command**: sudo apt upgrade
- **Description**: Upgrades all installed packages to the latest versions available in the repositories. This command will prompt you for confirmation before proceeding with the upgrades.

Update Raspberry Pi Firmware

✓ **Update Firmware**

- **Command**: sudo rpi-update
- **Description**: Updates the firmware of the Raspberry Pi. Use this command with caution, as it installs the latest firmware, which may include experimental changes.

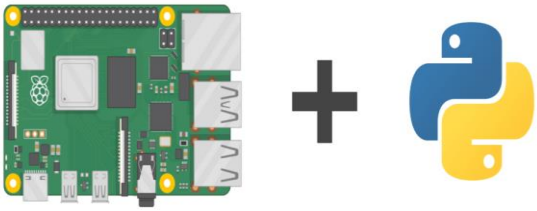


Chapter 4: Python Programming on Raspberry Pi

[Raspberry Pi Text Book]

Python is a versatile and beginner-friendly programming language that is ideal for use on the Raspberry Pi. This chapter will introduce you to setting up Python, writing your first Python script, and exploring some basic projects and libraries.

The main reason why Python is used on Raspberry Pi is that it's easy to use. The Raspberry Pi Foundation's goal is to help young students learn how to code, and using a simple language like Python is essential. Python is also powerful, with tons of libraries readily available.



1. Setting Up Python on Raspberry Pi

- **Checking Python Installation**

❑ **Verify Python Installation**:

•Python comes pre-installed on Raspberry Pi OS. To check the installed version, open a terminal and type:

```
python3 --version
```

•You should see the installed version of Python 3.

- **Install Python (if not installed)**:

❑ If Python is not installed, you can install it using:

```
sudo apt update sudo apt install python3
```

2. Installing Python Packages

❑ **Using pip**:

•pip is the package installer for Python. To install pip, use:

```
sudo apt install python3-pip
```

3. Writing Your First Python Script

- **Creating a Python Script**:

❑ Open a text editor like Nano:

```
nano hello.py
```

2.Writing the Script:

❑ Type the following code to print "Hello, World!":

```
print("Hello, World!")
```

3.Running the Script:

❑ Save the script by pressing Ctrl+X, Y, and Enter.

❑ Run the script using Python:

```
python3 hello.py
```

The Raspberry Pi's General Purpose Input/output (GPIO) pins are a powerful feature that allows you to interface with the physical world. This chapter will cover the basics of using GPIO pins, including how to connect and control hardware components such as LEDs, buttons, and sensors.

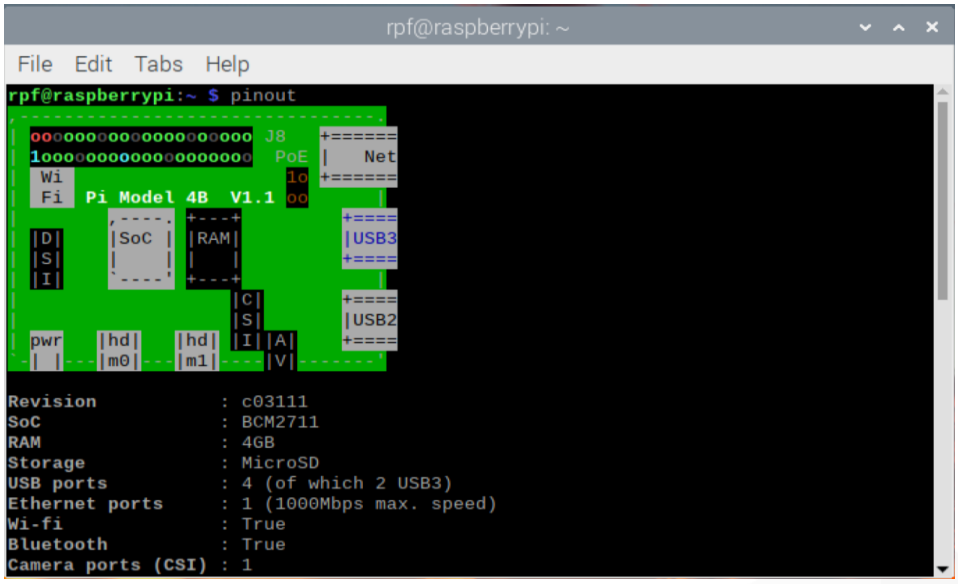
1. Introduction to GPIO Pins

The Raspberry Pi has a set of GPIO pins that can be used to interface with various electronic components. These pins are located on the GPIO header and can be programmed to perform different tasks.

- **GPIO Pinout**

1.Pin Layout:

1. The GPIO header has 40 pins (on Raspberry Pi models with 40-pin headers), which include power pins, ground pins, and GPIO pins.
2. GPIO pins can be configured as input or output.
3. Power pins provide 3.3V or 5V power.
4. Ground pins are used to complete electrical circuits.



1.Pin Numbering:

1. **BCM (Broadcom) Numbering**: Refers to the pin numbers defined by the Broadcom SOC (System on Chip).
2. **Board Numbering**: Refers to the physical pin numbers on the GPIO header.

Setting Up the GPIO Library

To control the GPIO pins using Python, you'll need the RPi.GPIO library, which comes pre-installed on Raspberry Pi OS.

1. Importing the RPi.GPIO Library
1.Import Statement:

```
import RPi.GPIO as GPIO
```

2.Setting Up the Mode:

•You can set up the mode to use either BCM or Board numbering.

```
GPIO.setmode(GPIO.BCM) # or GPIO.setmode(GPIO.BOARD)
```

2. Basic GPIO Operations
1.Configuring a Pin as Output:

```
GPIO.setup(18, GPIO.OUT) # Pin 18 set as output
```

2.Configuring a Pin as Input:

```
GPIO.setup(23, GPIO.IN) # Pin 23 set as Input
```

3.Writing to an Output Pin:

```
GPIO.output(18, GPIO.HIGH) # Turn on GPIO.output(18, GPIO.LOW) # Turn off
```

4.Reading from an Input Pin:

```
input_state = GPIO.input(25)  #Reading Digital Value from pin 25 and stores
.
```

5.Cleaning Up:

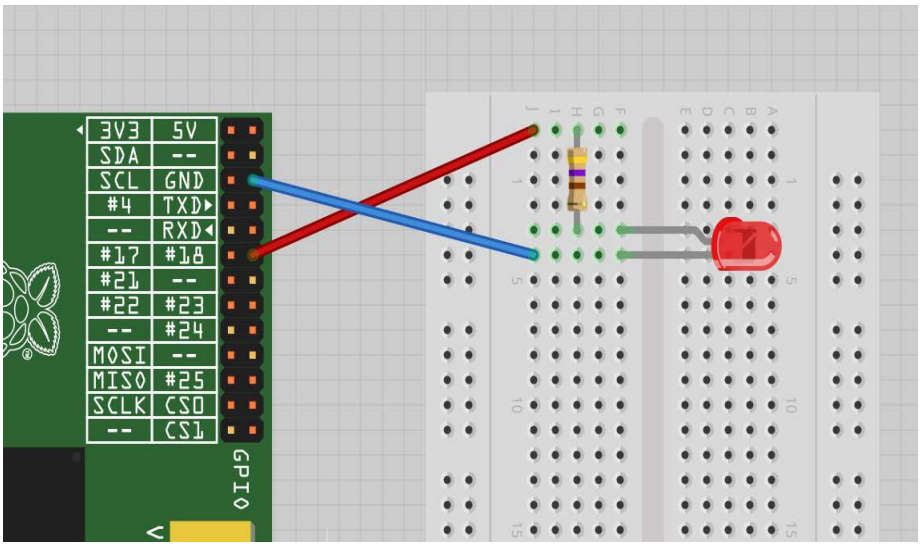
•It's important to clean up the GPIO settings after your script finishes running.

```
GPIO.cleanup()
```

1. Connecting an LED

Connect an LED to one of the GPIO pins using a 470Ω or 1kΩ series resistor to limit the current.

- Breadboard and jumper wires
- 1kΩ resistor
- LED



Having connected the LED, we need to be able to turn it on and off using commands from Python.
Start a Python console from the Terminal with superuser access and enter these commands:

```
$ sudo python
>>> import RPi.GPIO as GPIO
>>> GPIO.setmode(GPIO.BCM)
>>> GPIO.setup(18, GPIO.OUT)
>>> GPIO.output(18, True)
>>> GPIO.output(18, False)
```

This will turn your LED on and off.

✓ **LED Blinking**

Open Thonny IDE and create a file named blink.py and write the code given below with indentation.

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
   while (True):
        GPIO.output(18, True)
        time.sleep(0.5)
        GPIO.output(18, False)
        time.sleep(0.5)
```

The Led connected to Pin 18 is now starts Blinking.

Task : Try to blink Leds connected to other GPIO Pins.

2. Controlling the Brightness of an LED

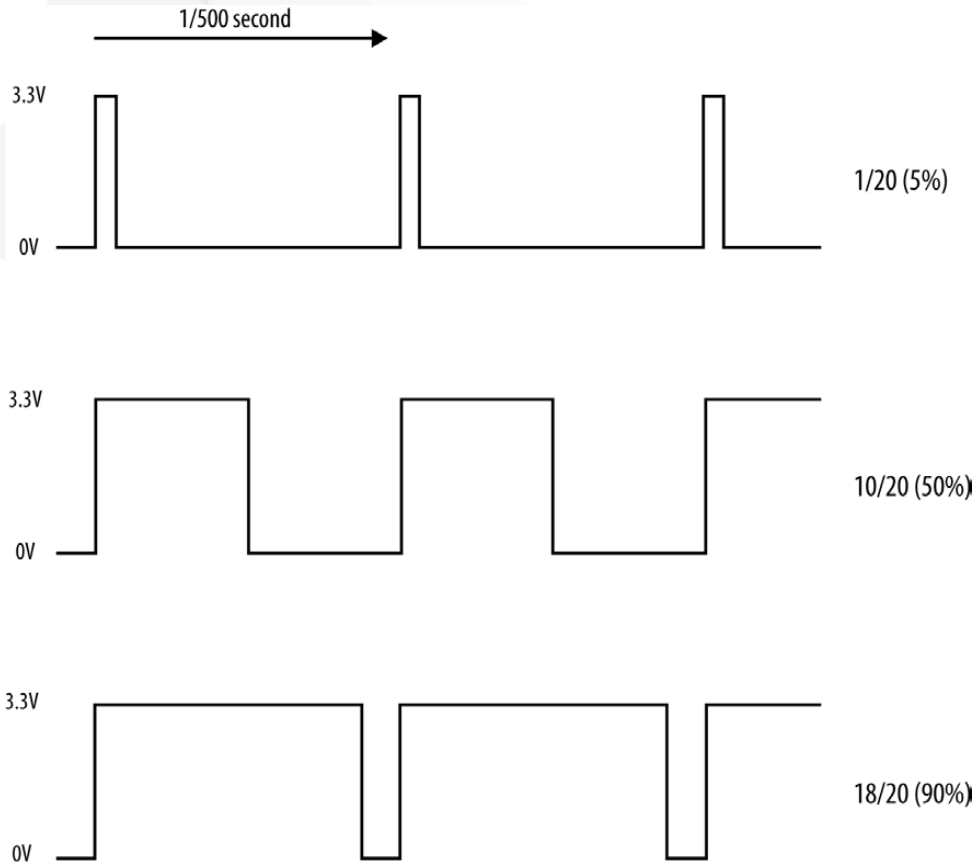
The RPi.GPIO library has a pulse-width modulation (PWM) feature that allows you to control the power to an LED and its brightness.
To try it out, connect an LED to Pin 18 and run this test program:

```
import RPi.GPIO as GPIO
led_pin = 18
GPIO.setmode(GPIO.BCM)
GPIO.setup(led_pin, GPIO.OUT)
pwm_led = GPIO.PWM(led_pin, 500)
pwm_led.start(100)
while True:
  duty_s = input("Enter Brightness (0 to 100):")
  duty = int(duty_s)
  pwm_led.ChangeDutyCycle(duty)
```

Run the Python program, and you will be able to change the brightness by entering a number between 0 and 100:

You can exit the program by pressing Ctrl-C.

PWM is a clever technique where you vary the length of pulses while keeping the overall number of pulses per second (the frequency in Hz) constant. Figure illustrates the basic principle of PWM.



At high frequencies, the measured PWM frequency varies somewhat from the frequency supplied as an argument. This may be something that changes in later versions of the PWM feature of RPi.GPIO.

You can change the PWM frequency by modifying this line:

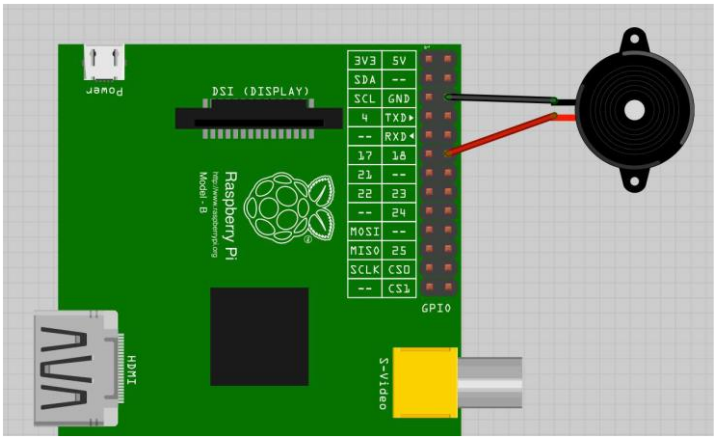
```
pwm_led = GPIO.PWM(led_pin, 500)
```

The value is in Hz, so in this case, the frequency is set to 500 Hz.

3. Make a Buzzing Sound

Use a piezo-electric buzzer connected to a GPIO pin.
Most small piezo buzzers work just fine using the arrangement shown in Figure .

You can connect the buzzer pins directly to the Raspberry Pi using female-to-female headers

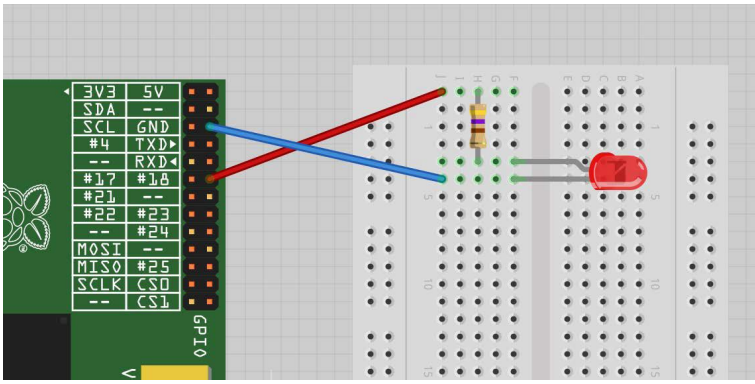


These buzzers use very little current. However, if you have a large buzzer or just want to play it safe, then put a 470Ω resistor between the GPIO pin and the buzzer lead.

```
import RPi.GPIO as GPIO
import time
buzzer_pin = 18
GPIO.setmode(GPIO.BCM)
GPIO.setup(buzzer_pin, GPIO.OUT)
def buzz(pitch, duration):
        period = 1.0 / pitch
        delay = period / 2
        cycles = int(duration * pitch)
for i in range(cycles):
        GPIO.output(buzzer_pin, True)
        time.sleep(delay)
        GPIO.output(buzzer_pin, False)
        time.sleep(delay)
while True:
        pitch_s = input("Enter Pitch (200 to 2000): ")
        pitch = float(pitch_s)
        duration_s = input("Enter Duration (seconds): ")
         duration = float(duration_s)
        buzz(pitch, duration)
```

4. Making a User Interface to Turn Things On and Off

Using the Tkinter user interface framework of Python , we can control the led connected in first project .It uses a check button to turn the GPIO pin on and off.



Tkinter is the standard GUI (Graphical User Interface) library for Python. It is included with Python, making it a convenient choice for creating desktop applications.

```
from Tkinter import * # Imports all classes and functions from the Tkinter module
import RPi.GPIO as GPIO # Imports the RPi.GPIO library and renames it to GPIO
import time # Imports the time module

GPIO.setmode(GPIO.BCM) # Sets the pin numbering system to BCM
GPIO.setup(18, GPIO.OUT) # Sets GPIO pin 18 as an output pin

class App:
    def __init__(self, master):
        frame = Frame(master)  # Create a frame widget as a container
        frame.pack()  # Pack the frame into the main window
        self.check_var = BooleanVar()  # Create a Boolean variable to store the checkbox state
        check = Checkbutton(frame, text='Pin 18', command=self.update, variable=self.check_var,
onvalue=True, offvalue=False)
        check.grid(row=1)  # Place the checkbox in the frame using grid layout

    def update(self):
        GPIO.output(18, self.check_var.get())  # Set GPIO pin 18 state to the checkbox state
root = Tk()  # Creates the main window
root.wm_title('On / Off Switch')  # Sets the window title
app = App(root)  # Creates an instance of the App class, passing the main window as the master
root.geometry("200x50+0+0")  # Sets the window size to 200x50 pixels and positions it at the top-
left corner of the screen
root.mainloop()  # Starts the Tkinter main loop, which waits for user interaction
```

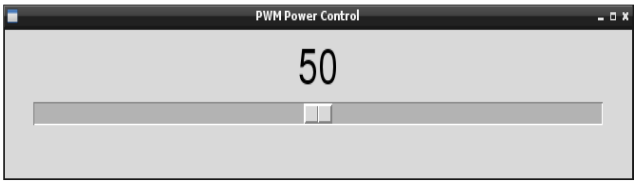
After running the program, a widget is created . Now , you can Control the led or device connected to that pin.

The example program defines a class called App that contains most of the application code. Its initializer function creates a member variable called check_var that contains an instance of BooleanVar that is then supplied as the variable option to the checkbutton.

This ensures that every time the checkbutton is clicked, the value in this variable will be changed. The command option runs the update command every time such a change occurs.
The update function simply writes the value in check_var to the GPIO output.

5. Making a User Interface to Control PWM Power for LEDs and Motors

Using the Tkinter user interface framework, write a Python program that uses a slider to change the PWM duty cycle between 0 and 100 percent.

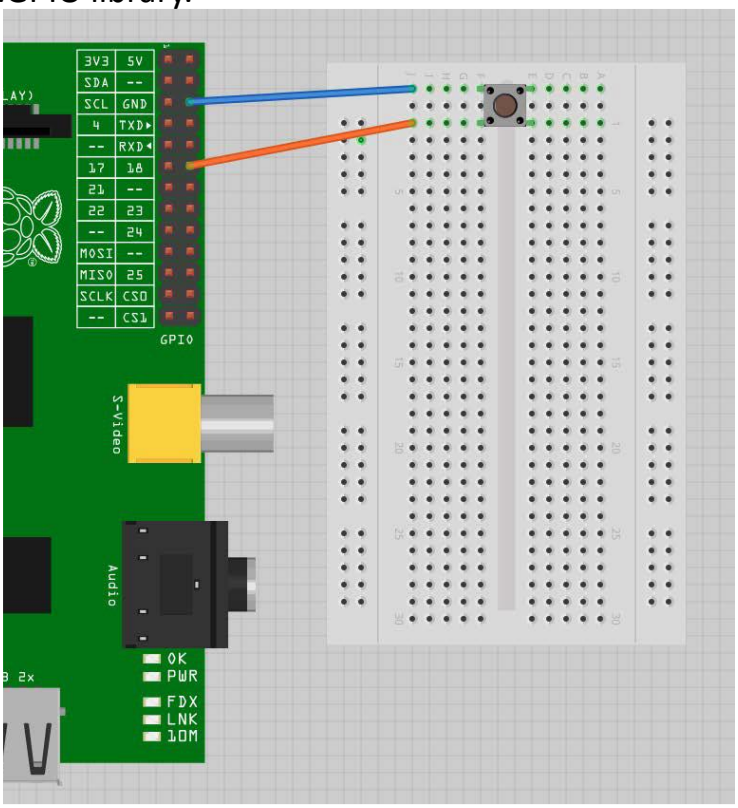



```
from Tkinter import *
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
pwm = GPIO.PWM(18, 500)
pwm.start(100)
class App:def __init__(self, master):
        frame = Frame(master)
        frame.pack()
        scale = Scale(frame, from_=0, to=100,
        orient=HORIZONTAL,
        command=self.update)
        scale.grid(row=0)
def update(self, duty):
       pwm.ChangeDutyCycle(float(duty))
root = Tk()
root.wm_title('PWM Power Control')
app = App(root)
root.geometry("200x50+0+0")
root.mainloop()
```

6. Programming with Interrupts

You want to respond to some event, such as a button push, without having to continually poll the input pin to see if its state has changed.
Use the add_event_detect function of the RPi.GPIO library.

This example code continually updates a count in seconds and displays a message when the button is pressed:




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

def my_callback(channel):
    print('You pressed the button')

GPIO.setup(18, GPIO.IN,
pull_up_down=GPIO.PUD_UP)
GPIO.add_event_detect(18, GPIO.FALLING,
callback=my_callback)

i = 0
while True:
    i = i + 1
    print(i)
    time.sleep(1)
```

Output

```
1
2
3
You pressed the button
4
You pressed the button
5
You pressed the button
You pressed the button
6
```

7. Web Interface

Bottle is a lightweight and flexible web framework for Python. It is designed to be simple and fast, making it a great choice for small web applications, prototyping, and learning web development. Here's an introduction to Bottle:

Introduction to Bottle

What is Bottle?

•**Bottle** is a micro web framework for Python.
•It is extremely lightweight and is distributed as a single file module with no dependencies other than the Python Standard Library.
•Bottle is designed to be simple and fast, making it ideal for small applications, APIs, and prototyping.
on the Python Standard Library.

Key Features of Bottle

1.Single-File Distribution:
- ✓ Bottle consists of a single file that you can easily include in your project.

2.Routing:
- ✓ Supports dynamic URL routes and RESTful request dispatching.

3.Templates:
- ✓ Built-in support for templates using a simple templating engine.

4.Utilities:
- ✓ Provides utilities for convenient access to form data, file uploads, cookies, headers, and other HTTP-related data.

5.Server Adapters:
- ✓ Compatible with multiple WSGI (Web Server Gateway Interface) servers and comes with built-in HTTP development server.

6.No Dependencies:
- ✓ Only depends

Getting Started with Bottle

Installation

You can install Bottle using pip:

```
pip install bottle
```

Basic Example

Here's a basic example of a Bottle application:

```
from bottle import route, run, template

@route('/')
def home():
    return template('<b>Hello {{name}}</b>!', name='World')

@route('/hello/<name>')
def hello(name):
    return template('<b>Hello {{name}}</b>!', name=name)

if __name__ == '__main__':
    run(host='localhost', port=8080)
```

Explanation

•@route('/'): This decorator is used to bind a function to a URL path. In this case, the home function is bound to the root URL (/).

•template('Hello {{name}}!', name='World'): Renders a template with a placeholder replaced by the value of name.

•@route('/hello/<name>'): This route captures a dynamic part of the URL and passes it to the function as an argument.

•run(host='localhost', port=8080): Starts the development server on localhost at port 8080.

Key Concepts

1.Routing: Mapping URLs to functions.

2.Templates: Using templates to generate HTML dynamically.

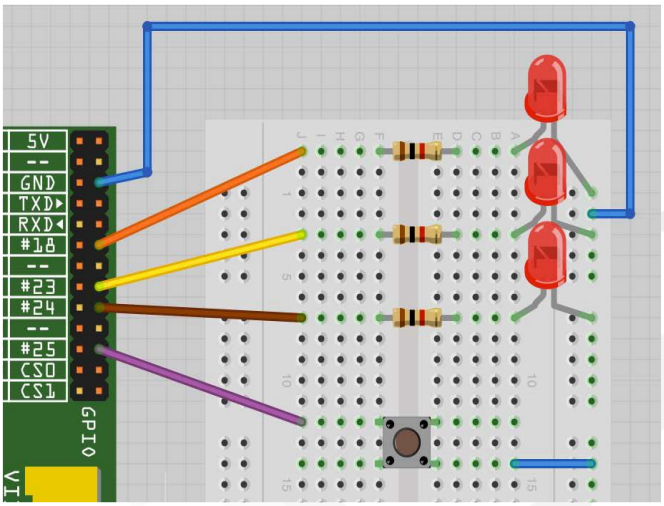
3.Forms and Request Data: Handling form data and other HTTP request data.

4.File Uploads: Handling file uploads from HTML forms.

5.Static Files: Serving static files like images, CSS, JavaScript, etc.

Project : Controlling GPIO Outputs Using a Web Interface

In this project we will create a Web Application using bottle frame work With HTML coding with a little bit of JavaScript code to control the leds and Know the status of button.




```
from bottle import route, run
import RPi.GPIO as GPIO

host = '192.168.1.8'

 # GPIO setup
GPIO.setmode(GPIO.BCM) # Use Broadcom GPIO numbering
led_pins = [18, 23, 24] # List of GPIO pins connected to LEDs
led_states = [0, 0, 0] # Initial states of the LEDs (off)
switch_pin = 25 # GPIO pin connected to the switch

# Setup LED pins as output
for pin in led_pins:
   GPIO.setup(pin, GPIO.OUT)
```

```
 # Setup switch pin as input with pull-up resistor
 GPIO.setup(switch_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
def switch_status():
    """Check the status of the switch and return 'Up' or 'Down'."""
    state = GPIO.input(switch_pin)
    return 'Up' if state else 'Down'

def html_for_led(led):
 """Generate HTML for a button to control an LED."""
return f"<input type='button' onClick='changeLed({led})' value='LED {led}'/>"
def update_leds():
    """Update the GPIO pins according to the current LED states."""
    for i, state in enumerate(led_states):
        GPIO.output(led_pins[i], state)

@route('/')
@route('/<led>')
 def index(led="n"):
     """Render the web page and handle LED control."""
     if led != "n":
         # Convert LED number from URL to integer and toggle its state
          led_num = int(led)
          led_states[led_num] = not led_states[led_num]
          update_leds()

# Build the HTML response
     response = "<script>"
      response += "function changeLed(led) {"
      response += " // Redirect to URL with LED number"
      response += " window.location.href = '/' + led + ';"
      response += " // Reload the page to reflect the updated LED state"
      response += " setTimeout(function() { window.location.reload(); }, 100);"
      response += "}" response += "</script>" response += '<h1>GPIO Control</h1>'
      response += f'<h2>Button={switch_status()}</h2>'
      response += '<h2>LEDs</h2>'
      response += html_for_led(0)
      response += html_for_led(1)
      response += html_for_led(2)
      return response
# Run the Bottle web server
run(host=host, port=80)
```

Note

✓ Change the host address in code '192.168.1.8' to your device ip address.
To know the ip address type ip a in terminal.

```
pi@raspberrypi:~ $ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether dc:a6:32:a3:34:89 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.179/24 brd 192.168.1.255 scope global dynamic noprefixroute eth0
       valid_lft 4097sec preferred_lft 3197sec
    inet6 fe80::254:61c5:bad2:343f/64 scope link
       valid_lft forever preferred_lft forever
3: wlan0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN group default qlen 1000
    link/ether dc:a6:32:a3:34:8a brd ff:ff:ff:ff:ff:ff
pi@raspberrypi:~ $
```

Output