2

PROGRAMMING RASPBERRY PI

Unit Structure

2.0 OBJECTIVES

After going through this unit, you will be able to:

- Understand the concept of Raspberry Pi and its configuration with Linux commands
- Acquaint with the programming interfaces such as Node.js and Python
- Understand and implement various Raspberry Pi interfaces
- Get familiar with useful case study implementations

2.1 INTRODUCTION

One of the most popular physical computing boards on the market is the Raspberry Pi. People use the Raspberry Pi every day to engage with the world around them, from hobbyists making DIY projects to students learning to program for the first time. The Raspberry Pi is a fantastic single-board computer (SBC) that can run Linux and a variety of other programs. Python is a user-friendly programming language that can be used in schools, web development, scientific research, and a variety of other fields. Python is pre-installed on the Raspberry Pi, so you may use it to create your own Raspberry Pi projects.

When it comes to dealing with the Raspberry Pi, you have various alternatives. The Pi is most typically used as a standalone computer, which necessitates the use of a monitor, keyboard, and mouse (listed below). The Pi can also be used as a headless computer to save money (without a monitor, keyboard, and mouse). Because you'll need to use a command-line interface (CLI) from another computer, this configuration has a slightly steeper learning curve.

The Raspberry Pi is a single-board computer created by the Raspberry Pi Foundation, a non-profit organization based in the United Kingdom. Its compact size, full Linux environment, and general-purpose input–output (GPIO) pins have gained it a significant following in the maker and DIY communities. It was originally meant to provide young people with an affordable computing option to learn how to program. With all of the features and capabilities crammed onto this compact board, the Raspberry Pi has no shortage of projects and applications.

People use the Raspberry Pi all across the world to learn programming, develop hardware projects, automate their homes, implement Kubernetes clusters and Edge computing, and even employ them in industrial applications. The Raspberry Pi is a low-cost computer that runs Linux and has a set of GPIO (general purpose input/output) ports for controlling electronic components and experimenting with the Internet of Things (IoT).

2.3 RASPBERRY PI AND LINUX

2.3.1 ABOUT RASPBIAN

The Raspberry Pi can run a variety of operating systems. While many Pi compatible OSes are Linux distributions (distros), the RasPi also supports Android, Chrome OS, and non-Linux images. Despite the numerous operating system options, the Raspberry Pi Foundation's own Raspberry Pi OS remains one of the best Raspberry Pi distros available. The operating system originally known as Raspbian, on the other hand, has evolved significantly since its start.

Raspbian is a free operating system based on Debian and designed specifically for the Raspberry Pi. An operating system is a collection of programs and tools that enable your Raspberry Pi to function. Raspbian, on the other hand, is more than just an operating system: it includes over 35,000 packages, which are pre-compiled software packages packaged in a convenient style for quick installation on your Raspberry Pi.

In June of 2012, the initial build of nearly 35,000 Raspbian packages, optimized for the Raspberry Pi, was completed. Raspbian, on the other hand, is still in active development, with the goal of enhancing the reliability and performance of as many Debian programs as possible. The Raspberry Pi Foundation is not linked with Raspbian. Raspbian was produced by a small, dedicated team of developers who are enthusiastic about the Raspberry Pi hardware, the Raspberry Pi Foundation's educational mission, and, of course, the Debian Project.

Raspbian is a Raspberry Pi operating system based on Debian. The Raspberry Pi Foundation has officially released it as the primary operating system for the line of Raspberry Pi single-board computers since 2015. Mike Thompson and Peter Green started Raspbian as an independent project. The first phase of construction was finished in June 2012. The operating system is currently being developed. The Raspberry Pi line's low-performance ARM CPUs are well-suited to Raspbian. As of the most recent release, Raspbian's main desktop environment is PIXEL, Pi Improved Xwindows Environment, and Lightweight. It consists of a modified LXDE desktop environment and the Openbox stacking window manager, which has been updated with a new theme and a few additional tweaks. As of the newest edition, the distribution includes a copy of the computer algebra tool Mathematica, a version of Minecraft dubbed Minecraft Pi, and a lightweight version of Chromium.

2.3.1.1 HISTORY OF RASPBIAN

The developers behind Raspbian have released several distinct versions since its beginning. Because it's a Linux-based distribution, it's simple to make changes and update it on a regular basis.

- **Raspbian Wheezy**

  The Raspberry Pi Foundation officially supported the first release of Raspbian in 2015, which was mostly based on Debian Wheezy. Wheezy is an unofficial copy of Debian Wheezy armhf, and before official support, Raspberry Pis came pre-installed with Debian Squeeze as the official operating system, which was later superseded by Raspbian Wheezy. This is because Wheezy's engineers noticed that Squeeze was being used to support less-capable ARM devices, causing the Pi's CPU to perform poorly during floating point-intensive applications like graphics programs.

- **Raspbian Jessie**

  Along with the usual security fixes and under-the-hood tweaks, Jessie added a few more obvious additions. The Raspberry Pi Foundation made some tiny tweaks to make it seem more like a'real' PC in order to make it not simply cheap computers for education, but also affordable computers in their own right. The LibreOffice suite and Claws Mail, for example, were installed as standard, allowing users to use word processors, spreadsheets, and email management from within Raspbian. For the first time, Raspberry Pi's booted to a Raspbian desktop GUI by default, rather than a Linux command line, as a result of a software update.

  Raspbian Jessie with PIXEL was released in September 2016 for people who wanted a GUI desktop. The PIXEL (Pi Improved Xwindow Environment, Lightweight) desktop was the first time the OS acquired a GUI desktop, as it had previously only been a Linux code screen - it even had a boot splash page like a genuine OS. Indicators of performance were also incorporated. When the Pi was overworked in previous versions, for example, red and yellow pixels would appear on the screen. Under voltage was indicated by a lightning bolt, and temperature warnings were indicated by a thermometer.

- **Raspbian Stretch**

  Debian releases new official distros every two years, and Raspbian, which has always been based on Debian, follows suit. Stretch was launched just before Jessie's two-year anniversary, and like Jessie before it, the changes to Stretch were designed to go unnoticed by the end user.

  The inbuilt Bluetooth audio manager, on the other hand, was one of the more noticeable changes. PulseAudio was used in Jessie, but it was replaced by bluez-alsa because the former was awkward and didn't do a good job of encoding diverse audio sources. Following the revelation of firmware vulnerability in the Pi 3 and Pi Zero W wireless chipsets, Stretch included a modification to the base code layer.

- **Raspbian Buster**

  Buster was released two years and one month after Stretch, and it corresponded with the release of the Raspberry Pi 4. With the exception of a few security updates, the organization conceded that there were "unfortunately" no significant functional differences between Buster and its predecessor. Buster, on the other hand, included a slew of improvements to the OS's overall appearance and feel, as well as tweaks to the user interface. The OS was given a flatter and cleaner look with this design upgrade, which provided the first major UI improvements since Jessie.

  Buster also replaced IDLE with the Thonny Python development environment as the default Python editor. This was accompanied by a number of modest functionality enhancements, such as the 'eject' symbol for deleting USB devices only appearing if there are devices to eject.

2.3.1.2 FEATURES OF RASPBIAN

The Raspberry Pi OS, like the Pi hardware, has grown significantly over time. Pi OS now supports both 32-bit and 64-bit images. Other Linux distributions for the Pi, such as Ubuntu, have 64-bit and 32-bit installers. The Raspberry Pi OS has gradually introduced more functions, with a focus on desktop use, which complements the new hardware. Whether as a desktop, network-attached storage (NAS) device, cluster, or something else, more RAM and a beefier processor combine with overlying software for an increasingly competent computing experience.

Programming resources have been built into the Raspberry Pi for quite some time now. Integrated development environments (IDEs) and office productivity tools such as the LibreOffice suite come pre-installed. A bookshelf app with access to a boatload of Raspberry Pi books and

publications, including MagPi and HackSpace, is now included. Additionally, a Magnifier software improves visibility for all users, especially for smaller on-screen objects, resulting in greater accessibility. It's in the section under Universal Access.

Basic features are as follows:

- Developer: Raspberry Pi Foundation
- OS family: Unix-like
- Source model: Open source
- Latest release: Raspbian Jessie with PIXEL / 16.02.2017
- Marketing target: Raspberry Pi
- Update method: APT
- Package manager: dpkg
- Platforms: ARM
- Kernel type: Monolithic
- Userland: GNU
- Default user interface: PIXEL, LXDE
- License: Free and open-source software licenses (mainly GPL)
- Official website: https://www.raspberrypi.org/downloads/raspbian/

2.3.1.3 WHO SHOULD USE THE RASPBERRY PI OPERATING SYSTEM?

Raspberry Pi OS is a fantastic desktop operating system. When paired with an 8GB Pi board or even a 4GB Pi, the 64-bit version should demonstrate the credit card-sized maker board's multitasking and general computing capabilities. Raspberry Pi OS may be easily adapted for certain use cases because it is based on Linux. For a Raspberry Pi NAS, you can install media server software like Plex, Emby, or Subsonic. Alternatively, for a home theatre PC, install Kodi and VLC (HTPC). It's ideal for office productivity, such as picture and audio editing, as well as programming.

And gaming is a lot of fun. Many games run natively on the Raspberry Pi, or you can use emulators like Retro Arch to run them. In general, Raspberry Pi OS is the best distribution for the majority of Pi users. It's a flexible operating system that will undoubtedly be polished and improved in the future. A desktop variation with suggested applications, a barebones desktop image, and a simple command-line only option are among the alternatives offered. You may also look at Ubuntu, which has images for both 32-bit and 64-bit Raspberry Pi's.

2.4 LINUX COMMANDS

The Linux command is a piece of software that comes with the Linux operating system. Commands can be used to complete all simple and complicated operations. On the Linux terminal, the commands are run. The

terminal is a command-line interface for interacting with the system, comparable to the Windows command prompt. In Linux, commands are case-sensitive.

In comparison to other operating systems like Windows and MacOS, Linux has a robust command-line interface. Through its terminal, we can perform both basic and complicated tasks. We can perform some fundamental operations such as creating, removing, and moving files. We can also execute complicated jobs including administrative chores (such as package installation and user administration), networking tasks (such as ssh connections), security tasks, and so forth.

Because it offers a variety of assistance features, the Linux terminal is a user-friendly terminal. To open the Linux terminal, press the "CTRL + ALT + T" keys together, then click the "ENTER" key to run a command.

The top 50 most commonly used Linux commands will be discussed in this topic, along with examples. These commands are beneficial to both beginners and professionals.

- **Linux directory commands**

1] pwd command: It is used for displaying the current working directory location.

Syntax: pwd

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ pwd
/home/javatpoint
```

2] mkdir command: Used for creating a new directory under any directory.

Syntax: mkdir <directory name>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ mkdir new_directory
javatpoint@javatpoint-Inspiron-3542:~$
```

3] rmdir Command: used for deleting a directory.

Syntax: rmdir <directory name>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ rmdir new_directory
javatpoint@javatpoint-Inspiron-3542:~$
```

4] ls Command: for displaying a list of content of a directory.

Syntax: ls

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ ls
a             Desktop           examples.desktop  Music         sample
Akash         Directory         hello.c           pico          snap
a.out         Documents         hello.i           Pictures      Templates
composer.phar Downloads         hello.o           project       Test.txt
Demo.sh       eclipse           hello.s           Public        Videos
Demo.txt      eclipse-installer index.html        Python
Demo.txt~     eclipse-workspace mail              Python-3.8.0
```

5] cd Command: for changing the current directory.

Syntax: cd <directory name>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ cd Desktop
javatpoint@javatpoint-Inspiron-3542:~/Desktop$
```

- **Linux File commands**

6] **Touch Command:** used for creating an empty files. Executing this command once will create multiple empty files.

Syntax:

touch <file name>

touch <file1> <file2> ....

Output:

```
javatpoint@javatpoint-Inspiron-3542:~/Newfolder$ touch Demo.txt
javatpoint@javatpoint-Inspiron-3542:~/Newfolder$ touch Demo1.txt Demo2.txt
javatpoint@javatpoint-Inspiron-3542:~/Newfolder$ ls
Demo1.txt  Demo2.txt  Demo.txt
```

7] **cat command:** This command is a multi-purpose utility in the Linux system. It is used for creating a file, displaying its content, copy the content of one file to another file, etc.

Syntax: cat [OPTION]... [FILE]..

To create a file, execute it as follows:

cat **> <file** name**>**

// Enter file content

Press "**CTRL+ D**" keys to save the file. To display the content of the file, execute it as follows:

cat **<file** name**>**

Output:

```
javatpoint@javatpoint-Inspiron-3542:~/Newfolder$ cat > Demo.txt
This is a text file.
javatpoint@javatpoint-Inspiron-3542:~/Newfolder$ cat Demo.txt
This is a text file.
```

8] **rm Command: This command is used for removing a file**

Syntax: rm <file name>

Output:

9] **cp command: It is used for copying a file or directory.**

Syntax: To copy in the same directory:

cp <existing file name> <new file name>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ cp demo.txt demo1.txt
javatpoint@javatpoint-Inspiron-3542:~$ cp demo.txt Documents
```

10] **mv Command: This command is used for moving a file or a directory from one location to another.**

Syntax: mv <file name> <directory path>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ mv demo.txt Directory
```

11] **rename Command: This command is used to rename the large group of files**

Syntax: rename 's/old-name/new-name/' files

Example: Execute the following command for converting the entire text files into pdf files

rename 's/\.txt$/\.pdf/' *.txt

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ rename 's/\.txt$/\.pdf/' *.txt
javatpoint@javatpoint-Inspiron-3542:~$ ls
a             Desktop           examples.desktop  Music      Python-3.8.0
Akash         Directory         hello.c           Newfolder  sample
a.out         Documents         hello.i           pico       snap
composer.phar Downloads         hello.o           Pictures   Templates
demo1.pdf     eclipse           hello.s           project    Test.pdf
Demo.sh       eclipse-installer index.html        Public     Videos
Demo.txt~     eclipse-workspace mail              Python
```

- **<u>Linux File Content Commands</u>**

12] **head Command: For** displaying the content of a file. It displays the first 10 lines of a file.

Syntax: head <file name>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ head Demo.txt
1
2
3
4
5
6
7
8
9
10
```

13] **tail Command: This is** similar to the head command. The only difference is this is used to display the last ten lines of the file content. It's useful for deciphering error messages.

Syntax: tail <file name>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ tail Demo.txt
2
3
4
5
6
7
8
9
10
11
```

14] **tac Command: This command is the** reverse of cat command, as its name specified. It reverses the order of the contents of the file (from the last line).

Syntax: tac <file name>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ tac Demo.txt
11
10
9
8
7
6
5
4
3
2
1
```

15] more command: The more command is quite similar to the cat command in that it displays the contents of a file in the same way that the cat command does. The only difference between the two methods is that the more command displays a screenful of output at a time in the event of larger files.

The following keys are used to scroll the page in the more command:

ENTER key: To scroll down page by line.

Space bar: To advance to the next page.

b key: To return to the previous page.

/ key: To search the string.

Syntax: more <file name>

Output:


```
;;; gyp.el - font-lock-mode support for gyp files.

;; Copyright (c) 2012 Google Inc. All rights reserved.
;; Use of this source code is governed by a BSD-style license that can be
;; found in the LICENSE file.

;; Put this somewhere in your load-path and
;; (require 'gyp)

(require 'python)
(require 'cl)

(when (string-match "python-mode.el" (symbol-file 'python-mode 'defun))
  (error (concat "python-mode must be loaded from python.el (bundled with "
                 "recent emacsen), not from the older and less maintained "
                 "python-mode.el")))

(defadvice python-indent-calculate-levels (after gyp-outdent-closing-parens
                                                 activate)
  "De-indent closing parens, braces, and brackets in gyp-mode."
  (when (and (eq major-mode 'gyp-mode)
             (string-match "^ *[])}][,)}]* *$"
                           (buffer-substring-no-properties
--More--(7%)
```

16] less Command: The less command works in the same way as the more command. It also has some added functions, such as 'terminal width and height modification.' The more command, on the other hand, reduces the output to the width of the terminal.

Syntax: less <file name>

Output:

```
;;; gyp.el - font-lock-mode support for gyp files.

;; Copyright (c) 2012 Google Inc. All rights reserved.
;; Use of this source code is governed by a BSD-style license that can be
;; found in the LICENSE file.

;; Put this somewhere in your load-path and
;; (require 'gyp)

(require 'python)
(require 'cl)

(when (string-match "python-mode.el" (symbol-file 'python-mode 'defun))
  (error (concat "python-mode must be loaded from python.el (bundled with "
                 "recent emacsen), not from the older and less maintained "
                 "python-mode.el")))

(defadvice python-indent-calculate-levels (after gyp-outdent-closing-parens
                                                 activate)
```

- **Linux User Commands**

17] su Command: The su command grants another user administrative privileges. In other words, it grants another user access to the Linux shell.

Syntax: su <user name>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ su javatpoint
Password:
javatpoint@javatpoint-Inspiron-3542:~$
```

18] id Command: used for displaying the user ID (UID) and group ID (GID).

Syntax: id

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ id
uid=1000(javatpoint) gid=1000(javatpoint) groups=1000(javatpoint),4(adm),24(cdro
m),27(sudo),30(dip),46(plugdev),116(lpadmin),126(sambashare)
javatpoint@javatpoint-Inspiron-3542:~$
```

19] useradd Command: On a Linux server, the useradd command is used to add or remove users.

Syntax: useradd username

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ sudo useradd JTP
[sudo] password for javatpoint:
javatpoint@javatpoint-Inspiron-3542:~$
```

20] passwd Command: The passwd command is used to set and update a user's password.

Syntax: passwd <username>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ sudo passwd JTP
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

21] groupadd Command: Used for create a user group.

Syntax: groupadd <group name>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ sudo groupadd Developer
javatpoint@javatpoint-Inspiron-3542:~$
```

- **Linux Filter Commands**

22] cat Command: The cat command can also be used to filter data. It's used inside pipes to filter files.

Syntax: cat <fileName> | cat or tac | cat or tac |. . .

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ cat Demo.txt | tac | cat | cat | tac
1
2
3
4
5
6
7
8
9
10
11
```

23] cut Command: To choose a specific column of a file, use the cut command. A space (' '), a slash (/), a hyphen (-), or anything else can be used as a delimiter using the '-d' option. A column number is specified using the '-f' option.

Syntax: cut -d(delimiter) -f(columnNumber) <fileName>

Output:


```
javatpoint@javatpoint-Inspiron-3542:~$ cat >marks.txt
alex-50
alen-70
jon-75
carry-85
celena-90
justin-80
javatpoint@javatpoint-Inspiron-3542:~$ cut -d- -f2 marks.txt
50
70
75
85
90
80
javatpoint@javatpoint-Inspiron-3542:~$
```

24] grep Command: In a Linux system, the grep command is the most powerful and often used filter. "Global regular expression print" is what grep stands for. It's useful for looking for information in a file. It's usually used in conjunction with a pipe.

Syntax: command | grep <searchWord>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ cat marks.txt | grep 9
celena-90
```

25] comm Command: To compare two files or streams, use the 'comm' command. It displays three columns by default: the first column shows non-matching things from the first file, the second column shows non-matching items from the second file, and the third column shows matched items from both files.

Syntax: comm <file1> <file2>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ comm Demo.txt Demo1.txt
                1
2
                3
comm: file 2 is not in sorted order
        11
                4
                5
        22
        33
6
7
8
9
comm: file 1 is not in sorted order
10
11
```

26] sed command: sed is also known as the stream editor command. It's used to employ a regular expression to modify files. It does not edit files indefinitely; instead, the modified material is just displayed. It has no effect on the file itself.

Syntax: command | sed 's/<oldWord>/<newWord>/'

Output:

27] tee command: The cat command and the tee command are very similar. The sole difference between the two filters is that one writes standard input to standard output while the other does not.

Syntax: cat <fileName> | tee <newFile> | cat or tac |.....

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ cat marks.txt | tee new.txt | cat
alex-50
alen-70
jon-75
carry-85
celena-90
justin-80
javatpoint@javatpoint-Inspiron-3542:~$ cat new.txt
alex-50
alen-70
jon-75
carry-85
celena-90
justin-80
```

28] tr Command: The tr command is used to convert file text from lower case to upper case, for example.

Syntax: command | tr <'old'> <'new'>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ cat marks.txt | tr 'prcu' 'PRCU'
alex-50
alen-70
jon-75
CaRRy-85
Celena-90
jUstin-80
```

29] uniq Command: The uniq command creates a sorted list in which each word appears just once.

Syntax: command <fileName> | uniq

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ sort marks.txt |uniq
alen-70
alex-50
carry-85
celena-90
jon-75
justin-80
```

30] wc Command: A file's lines, words, and characters are counted with the wc programme.

Syntax: wc <file name>

Output:


```
javatpoint@javatpoint-Inspiron-3542:~$ wc marks.txt
 6  6 52 marks.txt
```

31] od Command: The od command displays a file's content in various formats, including hexadecimal, octal, and ASCII characters.

Syntax:

od -b <fileName> // Octal format

od -t x1 <fileName> // Hexa decimal format

od -c <fileName> // ASCII character format

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ od -b marks.txt
0000000 141 154 145 170 055 065 060 012 141 154 145 156 055 067 060 012
0000020 152 157 156 055 067 065 012 143 141 162 162 171 055 070 065 012
0000040 143 145 154 145 156 141 055 071 060 012 152 165 163 164 151 156
0000060 055 070 060 012
0000064
javatpoint@javatpoint-Inspiron-3542:~$ od -t x1 marks.txt
0000000 61 6c 65 78 2d 35 30 0a 61 6c 65 6e 2d 37 30 0a
0000020 6a 6f 6e 2d 37 35 0a 63 61 72 72 79 2d 38 35 0a
0000040 63 65 6c 65 6e 61 2d 39 30 0a 6a 75 73 74 69 6e
0000060 2d 38 30 0a
0000064
javatpoint@javatpoint-Inspiron-3542:~$ od -c marks.txt
0000000   a   l   e   x   -   5   0  \n   a   l   e   n   -   7   0  \n
0000020   j   o   n   -   7   5  \n   c   a   r   r   y   -   8   5  \n
0000040   c   e   l   e   n   a   -   9   0  \n   j   u   s   t   i   n
0000060   -   8   0  \n
0000064
```

32] sort Command: Sorting files in alphabetical order is done with the sort command.

Syntax: sort <file name>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ sort marks.txt
alen-70
alex-50
carry-85
celena-90
jon-75
justin-80
```

33] gzip Command: To reduce the file size, use the gzip command. It's a tool for compressing data. The compressed file with the '.gz' extension replaces the original file.

Syntax: gzip <file1> <file2> <file3>...

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ gzip Demo.txt Demo1.txt
javatpoint@javatpoint-Inspiron-3542:~$ ls
a              Demo.txt.gz         examples.desktop  Music      Python-3.8.0
Akash          Desktop             hello.c           Newfolder  sample
a.out          Directory           hello.i           new.txt    snap
composer.phar  Documents           hello.o           pico       Templates
demo1.pdf      Downloads           hello.s           Pictures   Test.pdf
Demo1.txt.gz   eclipse             index.html        project    Videos
Demo.sh        eclipse-installer   mail              Public
Demo.txt~      eclipse-workspace   marks.txt         Python
```

34] gunzip Command: To decompress a file, use the gunzip command. It's the inverse of the gzip command.

Syntax: gunzip <file1> <file2> <file3>. .

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ gunzip Demo.txt Demo1.txt
javatpoint@javatpoint-Inspiron-3542:~$ ls
a              Demo.txt~          examples.desktop  Music      Python-3.8.0
Akash          Desktop            hello.c           Newfolder  sample
a.out          Directory          hello.i           new.txt    snap
composer.phar  Documents          hello.o           pico       Templates
demo1.pdf      Downloads          hello.s           Pictures   Test.pdf
Demo1.txt      eclipse            index.html        project    Videos
Demo.sh        eclipse-installer  mail              Public
Demo.txt       eclipse-workspace  marks.txt         Python
```

- **Linux Utility Commands**

35] find Command: The find command allows you to locate a specific file within a directory. It also allows you to search for files by name, type, date, and other criteria.

Following the find command, the following symbols are used:

(.) For the current directory

(/): for the root

Syntax: find . -name "*.pdf"

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ find . -name "*.pdf"
./Test.pdf
./Python-3.8.0/Doc/library/turtle-star.pdf
./Akash/Joomla/Origional Copy/Brochure-Joomla-2019.pdf
./Akash/Joomla/Origional Copy/Joomla-Guide-Final.pdf
./.local/share/Trash/files/2400966-250544e72f817db3bcef-1587140240830.pdf
./.local/share/Trash/files/2400966-3ad982eaa58c5d43fb53-1585763620407.pdf
find: ‘./.anydesk/incoming’: Permission denied
./Downloads/ConfirmationPage_20030070774.pdf
./demo1.pdf
find: ‘./.dbus’: Permission denied
find: ‘./.cache/dconf’: Permission denied
./Directory/demo.pdf
./Directory/demo2.pdf
./Directory/demo1.pdf
```

36] locate Command: The locate command allows you to look for a file by name. It works similarly to the locate command, with the exception that it runs in the background. The find command searches the file system, whereas the find command examines the database. It's quicker than using the find command. Keep your database up to date if you want to use the locates command to find the file.

Syntax: locate <file name>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ locate sysctl.conf
/etc/sysctl.conf
/etc/sysctl.d/99-sysctl.conf
/etc/ufw/sysctl.conf
/snap/core/8935/etc/sysctl.conf
/snap/core/8935/etc/sysctl.d/99-sysctl.conf
/snap/core/9066/etc/sysctl.conf
/snap/core/9066/etc/sysctl.d/99-sysctl.conf
/snap/core18/1705/etc/sysctl.d/99-sysctl.conf
/snap/core18/1754/etc/sysctl.d/99-sysctl.conf
/usr/share/doc/procps/examples/sysctl.conf
/usr/share/man/man5/sysctl.conf.5.gz
```

37] date Command: The date command is used to display information such as the date, time, and time zone.

Syntax: date

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ date
Fri May 22 21:51:05 IST 2020
```

38] cal Command: The cal function displays the calendar for the current month, with the current date highlighted.

Syntax: cal<

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ cal
      May 2020
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
```

39] sleep Command: The sleep command is used to keep the terminal awake for a set period of time. It takes time in seconds by default.

Syntax: sleep <time>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ sleep 4
```

40] time Command: The time command is used to show the amount of time it takes to perform a command.

Syntax: time

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ time

real    0m0.000s
user    0m0.000s
sys     0m0.000s
```

41] zcat Command: The compressed files are displayed using the zcat command.

Syntax: zcat <file name>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ ls
a              Demo.txt.gz        examples.desktop  Music      Python-3.8.0
Akash          Desktop            hello.c           Newfolder  sample
a.out          Directory          hello.i           new.txt    snap
composer.phar  Documents          hello.o           pico       Templates
demo1.pdf      Downloads          hello.s           Pictures   Test.pdf
Demo1.txt      eclipse            index.html        project    Videos
Demo.sh        eclipse-installer  mail              Public
Demo.txt~      eclipse-workspace  marks.txt         Python
javatpoint@javatpoint-Inspiron-3542:~$ zcat Demo.txt
1
2
3
4
5
6
```

42] df Command: The df command is used to display the file system's disc space use. It shows the number of used blocks, available blocks, and the mounted directory in the output.

Syntax: df

Output:


```
javatpoint@javatpoint-Inspiron-3542:~$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
udev             1931652        0   1931652   0% /dev
tmpfs             393260     1756    391504   1% /run
/dev/sda1      479668904 26471148 428762148   6% /
tmpfs            1966284   243536   1722748  13% /dev/shm
tmpfs               5120        4      5116   1% /run/lock
tmpfs            1966284        0   1966284   0% /sys/fs/cgroup
/dev/loop1        231936   231936         0 100% /snap/wine-platform-runtime/136
/dev/loop2        144128   144128         0 100% /snap/gnome-3-26-1604/98
/dev/loop4           384      384         0 100% /snap/gnome-characters/539
/dev/loop6        220160   220160         0 100% /snap/wine-platform-5-stable/4
/dev/loop5        164096   164096         0 100% /snap/gnome-3-28-1804/116
```

43] mount Command: The mount command is used to attach a file system from an external device to the system's file system.

Syntax: mount -t type <device> <directory>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=1931652k,nr_inodes=482913,mo
de=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmod
e=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=393260k,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relat
ime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
```

44] exit Command: The exit command in Linux is used to quit the current shell. It accepts a number as an argument and leaves the shell with a status number return.

Syntax: exit

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ exit
```

It will exit the terminal after pressing the ENTER key.

45] clear Command: To clear the terminal screen, use the Linux clear command.

Syntax: clear

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ ls
a              Demo.txt.gz       examples.desktop  Music      Python-3.8.0
Akash          Desktop           hello.c           Newfolder  sample
a.out          Directory         hello.i           new.txt    snap
composer.phar  Documents         hello.o           pico       Templates
demo1.pdf      Downloads         hello.s           Pictures   Test.pdf
Demo1.txt      eclipse           index.html        project    Videos
Demo.sh        eclipse-installer mail              Public
Demo.txt~      eclipse-workspace marks.txt         Python
javatpoint@javatpoint-Inspiron-3542:~$ clear
```

The terminal screen will be cleared after pressing the ENTER key.

- **Linux Networking Commands**

46] ip Command: The ipconfig command in Linux has been replaced by the ip command. Its functions include assigning an IP address, initialising an interface, and disabling an interface.

Syntax: ip a or ip addr

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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: enp7s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOW
N group default qlen 1000
    link/ether 74:e6:e2:02:93:b8 brd ff:ff:ff:ff:ff:ff
3: wlp6s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP gro
up default qlen 1000
    link/ether 00:71:cc:00:e2:89 brd ff:ff:ff:ff:ff:ff
    inet 192.168.43.240/24 brd 192.168.43.255 scope global dynamic noprefixroute
 wlp6s0
       valid_lft 2296sec preferred_lft 2296sec
    inet6 fe80::8c59:e84e:1670:27cc/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

47] ssh Command: The ssh command in Linux is used to establish a remote connection using the ssh protocol.

Syntax: ssh user_name@host(IP/Domain_name)</p>

48] mail Command: From the command line, the mail command is used to send emails.

Syntax: mail -s "Subject" <recipient address>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ mail -s "Hello World" Himanshudubey481@gmail.com
Cc:
Hello There
Hope you are doing well.
```

49] ping Command: The ping command is used to determine whether two nodes are connected, i.e. whether the server is connected. It's an abbreviation for "Packet Internet Groper."

Syntax: ping <destination>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ ping javatpoint.com
PING javatpoint.com (194.169.80.121) 56(84) bytes of data.
64 bytes from www.javatpoint.com (194.169.80.121): icmp_seq=1 ttl=48 time=3889 m
s
64 bytes from www.javatpoint.com (194.169.80.121): icmp_seq=2 ttl=48 time=3043 m
s
64 bytes from www.javatpoint.com (194.169.80.121): icmp_seq=3 ttl=48 time=2136 m
s
64 bytes from www.javatpoint.com (194.169.80.121): icmp_seq=4 ttl=48 time=1122 m
s
```

50] host Command: The host command displays the IP address associated with a specified domain name and vice versa. For the DNS Query, it does DNS lookups.

Syntax: host <domain name> or <ip address>

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ host javatpoint.com
javatpoint.com has address 194.169.80.121
```

2.5 CONFIGURING RASPBERRY PI WITH LINUX COMMANDS

When you first get your hands on a Raspberry Pi, you'll need to install an operating system and link it to a Micro-SD card. On the Raspberry Pi, Raspberry Pi not only supports their native Raspberry Pi OS, but also a variety of different Linux versions. So, once you've installed an operating system on a Raspberry Pi, you may communicate with it in a variety of ways.

- Using the HDMI connector to connect a monitor to experience a user interface
- Use the serial interface to communicate.
- **Remotely communicate using an SSH connection**

  When you connect to a display, you're faced with a user interface, and it's as simple as using your own computer to browse around the operating system. When connecting through serial interface or remote SSH, however, there is no such thing as a user interface. Instead, you'll have to use a command-line interface to navigate about your Raspberry Pi, which is analogous to the command prompt or PowerShell on a Windows PC and the terminal on a Macintosh.

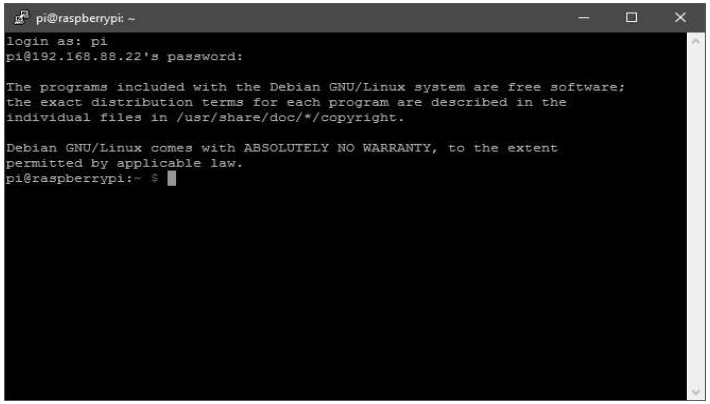
  When utilizing a command-line, you generally instruct the Raspberry Pi to complete tasks by typing instructions into the terminal, as opposed to the traditional method of interaction, which involves using a mouse. You might assume it's easier to communicate with a Raspberry Pi by connecting a display and using a user interface, but after you've mastered the command-line, your workflow will be much faster and you'll have more control over your Raspberry Pi. You'll be

able to combine these instructions into scripts and run them to speed up the completion of tasks. Also, there may be times when you need to deploy your Raspberry Pi to a different place, in which case the command line will come in handy.

This section will help you become familiar with the majority of the helpful commands you'll need to explore and interact with your Raspberry Pi! These commands will also work with any Linux distribution on the Raspberry Pi, as well as any other Linux-based system!

- **Command-line on the Raspberry Pi**

  The prompt pi@raspberrypi $ will appear on the first line when you log in to the command line on your Raspberry Pi. This signifies that you have logged in to your Raspberry Pi successfully. You can type your commands in front of this text in the commands line.



- **Updating the system**

  When you first switch on your Raspberry Pi, it's a good idea to update the operating system and its sources to the most recent version. You can do so by typing the commands below:

  sudo apt-get update

  sudo apt-get upgrade

  sudo apt-get dist-upgrade

  sudo rpi-update

  These commands can be used alone or in combination, as shown below:

sudo apt-get update && sudo apt-get upgrade && sudo apt-get dist-upgrade && sudo rpi-update

Note that you must type "sudo" at the start of each command to tell the Raspberry Pi that you are a "root" user. This enables you to use all of the commands available in Linux without any limitations.

- **Navigating through files and folders**

  To navigate through your files and directories, there are a few commands you can use:

  pwd: It stands for print working directory, and it tells you where you are in the directory tree.

  ls : displays a list of all the contents of the directory you're in.

  ls –l: lists all of the files in the directory you're in and gives you further information about them.

  cd: This command is used to return to the root directory. When you use “cd” with the name of another folder in the current directory, however, you will be switched to that directory.

  cd..: This command is used to return from one directory to another.

- **Performing file and folder operations**

  You may use following commands to execute tasks like creating new folders, copying, moving, and deleting files and directories.

  mkdir: for creating a new directory

  Example: mkdir pidir will create a new directory, with the label "pidir" as the name.

  cp: you can use this command to copy files from one directory to another.

  Example: cp /home/pi/new/file.txt /home/pi/project/, copies the file.txt from the /home/pi/new/ directory and pastes it into the /home/pi/project/ directory.

  mv: This will perform a cut-and-paste operation, moving the file from one directory to another. This command, on the other hand, can be used to rename file names in the same directory.

  Example: mv /home/pi/new/file.txt /home/pi/project/ will copy file.txt from /home/pi/new/ to /home/pi/project/.

  Example: mv oldproject.txt newproject.txt will rename the file from oldproject to newproject.

  rm: This is handy for deleting files that are no longer needed.

Example: rm testfile.txt will remove testfile.txt from its current directory.

clear: This clears all commands from the current screen and replaces it with a fresh one.

- **Creating a new file and editing the contents**

  You may want to alter the contents of a file, such as a text file, after you've created it. You might wish to use a command-line text editor like GNU Nano for this. By running the command below, you will be able to create a new file named newproject.txt or modify an existing file named newproject.txt, and you will be given the option to add content to it.

  nano newproject.txt

  By simply modifying the file format, such as newproject.py for python files and newproject.conf for configuration files, you may create or edit different types of files in the same way.

  You'll be able to use arrow keys to browse around the newproject.txt text file and type content inside it once it's been created. When you're finished, press Ctrl+x on your keyboard, then Y when it asks if you want to save it.

- **Raspberry Pi hardware information**

  You may need to check the hardware details on your Raspberry Pi from time to time and be unsure how to do so. Don't be concerned. To check all of the hardware details, use the instructions listed below.

  cat /proc/cpuinfo : displays information about the processor.

  cat /proc/meminfo : displays information on the Raspberry Pi's memory.

  cat /proc/partitions: shows the number and size of partitions on your SD card.

  cat /proc/version : tells you what Pi version you're running.

  vcgencmd measure_temp: displays the CPU temperature, which is crucial to check if you're running heavy programmes and want to keep an eye on the temperature.

  free -o -h: displays the amount of system memory available.

  top d1 : This command examines the CPU load and shows information for each core.

  df -h: This command can be used to determine how much free disc space your Raspberry Pi has.

  uptime: this shows how long the Raspberry Pi has been running as well as the load average.

- **Troubleshoot Raspberry Pi hardware**
  If you're searching for a report on how the Raspberry Pi's CPU and RAM are being used by running programs, run the following command.

  htop

  This will allow you to see if a specific app is running as well as determine which apps are slowing down your Raspberry Pi. You can close this window by using ctrl+c.

  Also, if you're having network problems, run the following command to see a list of all the networks to which you're connected.

  ifconfig

  If you're using Ethernet, look for the eth0 portion, and if you're using Wi-Fi, look for the wlan0 section. You may also check out your IP address.

- **Shutdown and restart your Raspberry Pi**

  You may use a few of instructions to shut down or restart your Raspberry Pi right away.

  sudo shutdown -h now: This will turn off your Raspberry Pi right away.

  However, if you want a schedule to shut down in 2 hours, for example, type the following command. -02:00 sudo shutdown

  sudo reboot : This will restart your Raspberry Pi right away.

2.6 SUMMARY

This unit made us familiar with the fundamentals required for programming Raspberry Pi. Starting with the operating system required by Raspberry Pi that is Raspbian we saw different Linux commands used for Raspberry Pi programming.

2.7 LIST OF REFERENCES

1) Mastering the Raspberry Pi, Warren Gay, Apress(2014)

2) https://mitu.co.in/wp-content/uploads/2017/09/03-Raspbian-Operating-System.pdf

3) https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners

4) https://www.geeksforgeeks.org/linux-commands/

5) https://www.javatpoint.com/nodejs-tutorial

6) https://www.tutorialspoint.com/nodejs/index.htm

7) https://www.w3schools.com/python/

8) https://docs.python.org/3/tutorial/

9) https://www.javatpoint.com/python-tutorial

10) https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html

11) https://www.ti.com/lit/ug/spruf95/spruf95.pdf?ts=1632378909735&ref_url=https%253A%252F%252Fwww.google.com%252F

12) https://www.seeedstudio.com/blog/2020/02/19/how-to-use-raspberry-pi-gpio-pins-python-tutorial/

13) https://www.ti.com/lit/an/slva704/slva704.pdf?ts=1632361805005&ref_url=https%253A%252F%252Fwww.google.com%252F

14) https://embetronicx.com/tutorials/tech_devices/i2c_1/

15) https://practicalee.com/spi/

16) http://events17.linuxfoundation.org/sites/events/files/slides/Shuah_Khan_cross_compile_linux.pdf

17) https://projects.raspberrypi.org/en/projects/getting-started-with-picamera

UNIT END EXERCISES

1] Write a note on Raspbian.

2] Explain the history and features of Raspbian.

3] Write a detailed note on different Linux commands.

4] State the various Linux commands for configuring the Raspberry Pi.

❄❄❄❄❄❄❄

3

PROGRAMING INTERFACES

Unit Structure

3.0 OBJECTIVES

After going through this unit, you will be able to:

- Acquaint with the programming concepts and its real-world applications
- To understand the fundamentals and applications of Node.js
- To introduce with the cores and significance of python and its applications in several domain

3.1 INTRODUCTION TO NODE.JS

Node.js is a cross-platform environment and framework for running JavaScript applications, and it's commonly used to build networking and server-side applications. Node.js is a cross-platform runtime environment and framework for executing JavaScript outside of the browser. It's used to

make server-side and network web applications. It's free to use and open source. It is available for download at https://nodejs.org/en/.

Node.js is a real-time online application framework that uses an event-driven architecture and a non-blocking Input/ Output API to improve throughput and scalability. The frameworks available for web development for a long time were all based on a stateless approach. A stateless model is one in which the data generated in one session (such as user settings and events) is not saved for use in a subsequent session with that user. It took a lot of effort to keep track of a user's session information between requests. However, with Node.js, web applications may now have real-time two-way connections, where both the client and the server can initiate communication and freely share data.

3.1.1 WHY SHOULD YOU USE NODE.JS?

Let's look at what makes this framework so popular. The majority of the applications were built using a stateless request-response framework over time. In these kinds of apps, it's up to the developer to make sure the correct code was written to keep the user's web session alive as they worked with the system.

You may now work in real-time and have two-way communication with Node.js web applications. The state is preserved, and the communication can be initiated by either the client or the server.

3.1.2 FEATURES OF NODE.JS

Let's take a look at some of Node.js' most important features.

1] Concurrent request processing is aided by asynchronous event-driven IO, which is undoubtedly Node.js' most compelling feature. This functionality essentially means that whenever Node receives a request for an Input /Output operation, it will do the action in the background while continuing to process other requests. This differs from other programming languages in several ways. The code below shows a simple example of this-

```
var fs = require('fs');
        fs.readFile("Sample.txt",function(error,data)
        {
            console.log("Reading Data completed");
    });
```

- The code line above examines reading a file named Sample.txt. In other programming languages, the next line of processing would take place only after the full file has been read.
- However, in the case of Node.js, the definition of the function (‘function(error,data)') is the most significant part of the code to pay attention to. A callback function is what this is called.
- So, in this case, the file reading activity will begin in the background. While the file is being read, other processing can take place at the same time. This anonymous function will be called whenever the file read process is complete, and the text "Reading Data done" will be written to the console log.

2]` The V8 JavaScript Runtime engine, which is also utilized by Google Chrome, is used by Node. Node features a wrapper for the JavaScript engine that speeds up the runtime engine and, as a result, the processing of requests within Node.

3] Concurrent request handling - Another important feature of Node is its ability to manage several connections with very little overhead in a single process.

4] JavaScript is used by the Node.js library, which is another crucial part of Node.js development. Because a large portion of the development community is already familiar with javascript, developing with Node.js becomes easier for those who are.

5] The Node.js framework has a thriving and active community. Because of the active community, major upgrades to the framework are always available. This ensures that the framework is always up to date with the current web development trends.

3.1.3 WHO MAKES USE OF NODE.JS?

Many significant corporations use Node.js. A couple of them are listed below.

- Paypal - A number of sites within Paypal have begun to migrate to Node.js.
- LinkedIn - LinkedIn's Mobile Servers, which run the iPhone, Android, and Mobile Web products, are powered by Node.js.
- Node.js, which has a half-billion instals, was used by Mozilla to support browser APIs.
- eBay's HTTP API service is hosted in Node.js.

3.1.4 WHEN SHOULD YOU USE NODE.JS?

1] Node.js is ideally suited for use in real-time streaming or event-based systems like Applications for chatting

2] Game servers - If you need a fast and high-performance server that can handle thousands of requests at once, this is the framework for you.

3] Good for collaborative workplaces - This is ideal for document management setups. Multiple persons will submit their documents and make frequent modifications by checking out and checking in documents in a document management environment. Because the

event loop in Node.js can be triggered anytime documents are modified in a document managed environment, its ideal for these setups.

4] Advertisement servers - You may receive thousands of requests to extract adverts from a central server, and Node.js is an excellent foundation for this.

5] Multimedia streaming servers - Another suitable scenario for Node is for multimedia streaming servers, where clients request various multimedia materials from the server.

When you require a lot of parallelism but not a lot of devoted CPU time, Node.js is a smart choice.

Best of all, because Node.js is based on JavaScript, it works best when creating client-side applications that use the same framework.

3.1.5 WHEN WILL YOU AVOID USING NODE.JS?

Node.js can be used in a variety of applications for different reasons. The only time it should not be used is when the program requires significant processing durations. Node is designed to run in a single thread. If an application is required to perform some lengthy calculations in the background, it will be unable to handle any further requests. As previously said, Node.js is best used when processing requires less devoted CPU time.

3.1.6 COMPONENTS OF NODE.JS

The figure below depicts several key components of Node.js:

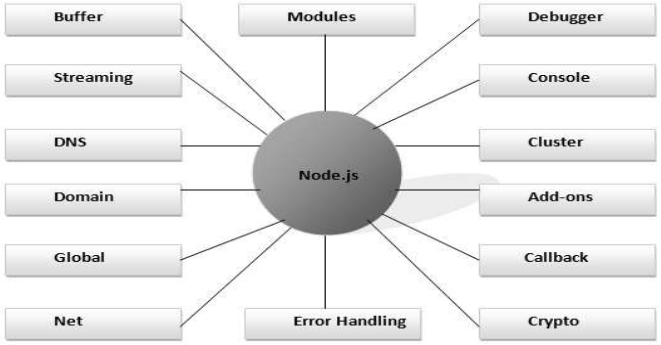


Figure 1.1 Components of Node.js

3.1.7 NODE.JS FRAMEWORKS AND TOOLS

Node.js is a low-level programming language. Thousands of libraries were written on Node.js by the community to make things easier and more exciting for developers. Many of these have become popular options over time. The following is a partial list of the ones worth learning:

- **AdonisJS:** It is a TypeScript-based, full-featured framework that prioritizes developer comfort, stability, and confidence. Adonis is a Node.js web framework that is one of the quickest.
- **Egg.js:** It is a framework that uses Node.js and Koa to create better enterprise frameworks and apps.
- **Express:** It's one of the simplest yet most powerful ways to set up a web server. Its success is due to its minimalist approach, which is unprejudiced and focused on the essential qualities of a server.
- **Fastify:** It is a web framework that focuses on giving developers the best possible experience with the least amount of overhead and a flexible plugin architecture. Fastify is a Node.js web framework that is one of the fastest.
- **FeatherJS:** It is a lightweight web framework that uses JavaScript or TypeScript to create real-time apps and REST APIs. Prototypes may be created in minutes, and production-ready apps can be developed in days.
- **Gatsby:** It is a static site generator built on React and powered by GraphQL, with a large ecosystem of plugins and starters.
- **Hapi:** It is a sophisticated framework for developing apps and services that allows developers to focus on defining reusable application logic rather than infrastructure.
- **koa:** It was created by the same team who created Express, and it strives to be even simpler and smaller, based on years of experience. The desire to make incompatible changes without disrupting the existing community spawned the new project.
- **Loopback.io:** Makes it simple to create modern apps with complex integrations.
- **Meteor** is a full-stack framework with an isomorphic approach to building apps using JavaScript that allows you to share code between the client and the server. Formerly an all-in-one tool, it now interfaces with the frontend libraries React, Vue, and Angular. It's also possible to make mobile apps with it.
- **Micro:** It creates asynchronous HTTP microservices using a very light server.
- **NestJS** is a TypeScript-based progressive Node.js framework for creating enterprise-grade server-side apps that are quick, dependable, and scalable.
- **Next.js:** A React framework with all the capabilities you need for production, including hybrid static and server rendering, TypeScript support, smart bundling, route pre-fetching, and more.

- **Nx:** It is a full-stack monorepo development toolkit that includes NestJS, Express, React, Angular, and more. Nx enables you to scale your development from a single team producing a single app to several teams working on multiple apps!
- **Sapper:** Sapper is a web application framework with a beautiful development experience and configurable filesystem-based routing for web applications of all sizes. Offers SSR as well as other services!
- **Socket.io:** It is a network application development platform that uses real-time communication.
- **Strapi:** Strapi is an open-source Headless CMS that allows developers to use their preferred tools and frameworks while also allowing editors to manage and distribute their content simply. Strapi helps the world's largest enterprises to expedite content delivery while creating stunning digital experiences by making the admin panel and API expandable through a plugin system.

3.2 PYTHON

Python is a dynamic, high-level, and interpreted programming language with a wide range of applications. It supports the development of applications using an Object-Oriented programming approach. It's simple and straightforward to learn, and it comes with a plethora of high-level data structures. Python is a scripting language that is simple to learn but powerful and versatile, making it ideal for application development. Python's syntax and dynamic typing, combined with the fact that it is interpreted, make it an excellent language for scripting and rapid application development. Python supports a variety of programming techniques, including object-oriented, imperative, functional, and procedural.

Python is not designed for a specific task, such as web programming. Because it can be used with web, enterprise, 3D CAD, and other applications, it is known as a multipurpose programming language. Because variables are dynamically typed, we don't need to use data types to declare them. For example, we can write a=10 to assign an integer value to an integer variable. Python allows for quick development and debugging because there is no compilation step in the development process, and the edit-test-debug cycle is very short.

3.2.1 PYTHON 2 Vs. PYTHON 3

When a new version of a programming language is released, it usually supports the features and syntax of the previous version, making it easier for projects to switch to the newer version. However, when it comes to Python, the two versions, Python 2 and Python 3, are vastly different.

The following is a list of differences between Python 2 and Python 3:

1] Print is a statement in Python 2 that can be used as print "something" to print a string to the console. Print, on the other hand, is a function in Python 3 that can be used as print("something") to print something to the console.

2] Raw input () is a function in Python 2 that accepts user input. It returns a string that represents the value entered by the user. To convert it into the integer, we need to use the int () function in Python. On the other hand, Python 3 uses input () function which automatically interpreted the type of input entered by the user. However, we can cast this value to any type by using primitive functions (int (), str (), etc.).

3] In Python 2, the implicit string type is ASCII, whereas, in Python 3, the implicit string type is Unicode.

4] The xrange() function from Python 2 is not available in Python 3. The xrange() function is a variant of the range() function that returns an xrange object that works in the same way as a Java iterator. The range() returns a list for example the function range(0,3) contains 0, 1, 2.

5] There is also a small change made in Exception handling in Python 3. It defines a keyword as which is necessary to be used. We will discuss it in Exception handling section of Python programming tutorial.

3.2.2 HISTORY OF PYTHON

Python was fabricated by Guido Van Rossum in 1991 at CWI in Netherland. The thought of Python programming language has taken from the ABCs programming language or we can say that ABCs may be a precursor of Python language. There is additionally a logic behind the selection of a name Python. Guido Van Rossum was an exponent of the popular BBC comedy show “Monty Python’s Flying Circus” at that era. Therefore, he decided to choose the name Python for his new created programming language. Python has the large community across the globe and releases its version inside the short amount.

3.2.3 WHY TO LEARN PYTHON?

Python is a scripting language that is high-level, interpreted, interactive, and object-oriented. Python is intended to be a very understandable language. It typically uses English terms instead of punctuation, and it has fewer syntactical structures than other languages.

Python is a must-have skill for students and working professionals who want to become exceptional software engineers, especially if they work in the Web Development field. Here some of the primary benefits of learning Python are discussed:

1] Python is Interpreted Python is handled by the interpreter during runtime. Before running your software, you do not need to assemble it. This is similar to the programming languages PERL and PHP.

2] Python is interactive in the sense that you can sit at a Python prompt and write your programs by interacting directly with the interpreter.

3] Python is Object-Oriented Python supports the Object-Oriented programming style or approach, which encapsulates code inside objects.

4] Python is a Fantastic Language for Beginners Python is a great language for beginners because it allows you to create a wide range of programs, from simple text processing to web browsers and games.

3.2.4 CHARACTERISTICS OF PYTHON

The following are some of the most important features of Python programming:

- It supports OOP as well as functional and structured programming methods.
- It can be used as a scripting language or compiled into byte-code for large-scale application development.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java

3.2.5 APPLICATIONS OF PYTHON

As mentioned before, Python is one of the most widely used languages over the web. Few of the applications are discussed here:

- Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- Easy-to-read − Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain − Python's source code is fairly easy-to-maintain.
- A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- Python is portable, meaning it can run on a wide range of hardware systems and has the same user interface across all of them.
- The Python interpreter can be extended by adding low-level modules. These modules allow programmers to improve the efficiency of their tools by adding to or customizing them.
- Python has interfaces to all of the major commercial databases.
- Python supports GUI applications that can be created and ported to a variety of system calls, libraries, and operating systems, including Windows MFC, Macintosh, and Unix's X Window system.
- Python is more scalable than shell scripting in terms of structure and support for large programs.

3.3 SUMMARY

Python is a scripting language that is high-level, interpreted, interactive, and object-oriented. Python is intended to be a very understandable language. It typically uses English terms instead of punctuation, and it has fewer syntactical structures than other languages.

Node.js (Node) is an open source server-side execution platform for JavaScript code. Node is commonly used for real-time applications like as chat, news feeds, and web push notifications and is useful for designing apps that require a persistent connection from the browser to the server.

3.4 LIST OF REFERENCES

1) Mastering the Raspberry Pi, Warren Gay, Apress(2014)
2) https://mitu.co.in/wp-content/uploads/2017/09/03-Raspbian-Operating-System.pdf
3) https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners
4) https://www.geeksforgeeks.org/linux-commands/
5) https://www.javatpoint.com/nodejs-tutorial
6) https://www.tutorialspoint.com/nodejs/index.htm
7) https://www.w3schools.com/python/
8) https://docs.python.org/3/tutorial/
9) https://www.javatpoint.com/python-tutorial
10) https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html
11) https://www.ti.com/lit/ug/spruf95/spruf95.pdf?ts=1632378909735&ref_url=https%253A%252F%252Fwww.google.com%252F
12) https://www.seeedstudio.com/blog/2020/02/19/how-to-use-raspberry-pi-gpio-pins-python-tutorial/
13) https://www.ti.com/lit/an/slva704/slva704.pdf?ts=1632361805005&ref_url=https%253A%252F%252Fwww.google.com%252F
14) https://embetronicx.com/tutorials/tech_devices/i2c_1/
15) https://practicalee.com/spi/
16) http://events17.linuxfoundation.org/sites/events/files/slides/Shuah_Khan_cross_compile_linux.pdf
17) https://projects.raspberrypi.org/en/projects/getting-started-with-picamera

3.5 UNIT END EXERCISES

1] Write a short note on Node.js.
2] Discuss the concept of Python.

❄❄❄❄❄❄❄

4

RASPBERRY PI INTERFACES

Unit Structure

4.0 OBJECTIVES

After going through this unit, you will be able to:

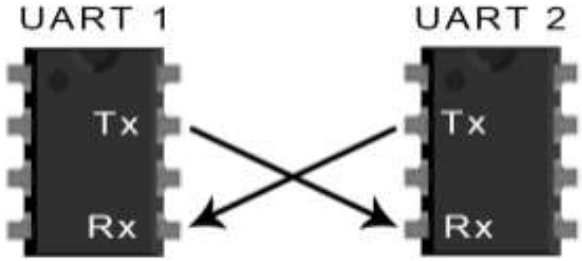
- Understand the concept and applications of communication interfaces
- Introduce with various raspberry pi communication interfaces such as UART, GPIO, I2C, SPI along with its characteristics, working and its applications point of view

4.1 UART

Universal Asynchronous Receiver/Transmitter (UART) is an acronym for Universal Asynchronous Receiver/Transmitter. It is a physical circuit in a microcontroller or a stand-alone IC, not a communication protocol like SPI or I2C. The primary function of a UART is to transmit and receive serial data.

4.1.1 INTRODUCTION TO UART COMMUNICATION

Two UARTs communicate directly with each other in UART communication. The transmitting UART translates parallel data from a controlling device, such as a CPU, into serial data and sends it to the receiving UART, which then converts the serial data back into parallel data for the receiving device. To send data between two UARTs, only two wires are required. Data transfers from the transmitting UART's Tx pin to the receiving UART's Rx pin:



UARTs send data asynchronously, which means there is no clock signal to synchronize the transmitting UART's output of bits with the receiving UART's sampling of bits. The transmitting UART adds start and stop bits to the data packet being transferred instead of a clock signal. These bits indicate the start and end of the data packet, allowing the receiving UART to determine when to begin reading the bits.

When a start bit is detected by the receiving UART, it begins reading the incoming bits at a particular frequency known as the baud rate. The baud rate is a unit of measurement for data transfer speed, given in bits per second (bps). Both UARTs must communicate at a similar baud rate. The baud rate

difference between the transmitting and receiving UARTs can only be about 10% before the bit timing becomes too off. Both UARTs must also be set up to send and receive data packets with the same structure.

4.1.2 WHY UART IS USED?

For quick communication, protocols such as SPI (serial peripheral interface) and USB (universal serial bus) are employed. UART is utilized when high-speed data transport is not necessary. It's a low-cost communication device that only has one transmitter and receiver. It only requires one wire for data transmission and another for data reception. An RS232-TTL or USB-TTL converter can be used to connect it to a PC (personal computer). The only thing that RS232 and UART have in common is that they both transmit and receive data without the use of a clock. For serial data transport, the UART frame comprises of one start bit, one or two stop bits, and a parity bit.

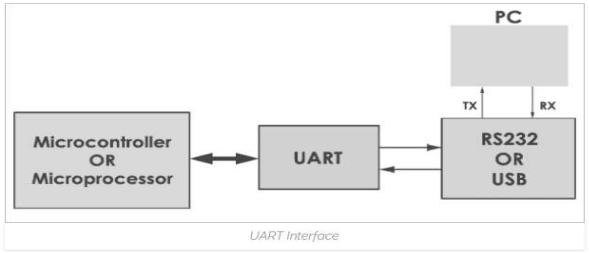


Figure 4.2 UART interface

4.1.3 BLOCK DIAGRAM

The fundamental components of the UART are as follows. They are the transmitter and the receiver, respectively. The Transmit hold register, Transmit shift register, and control logic make up the transmitter. A Receive hold register, Receiver shift register, and control logic are also present in the receiver. A baud rate generator is included in both the transmitter and the receiver.

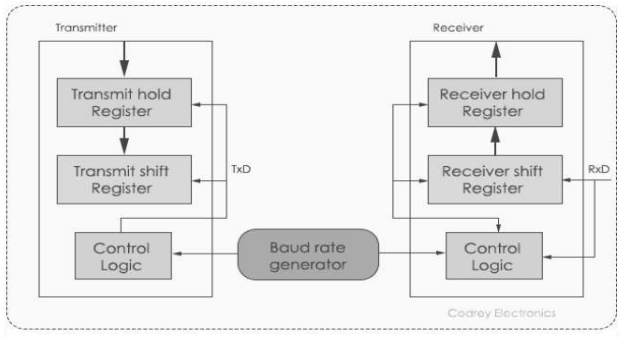


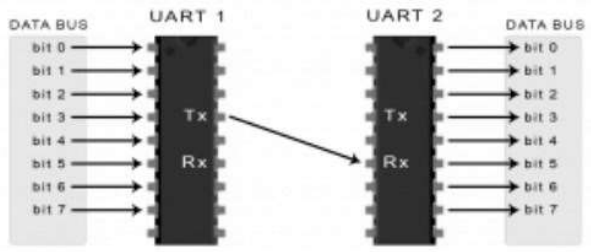
Figure 4.3 UART block diagram

The baud rate generator determines how fast the transmitter and receiver must send and receive data. The data byte to be transmitted is stored in the Transmit hold register. The bits are shifted to the left or right in the transmit and receive shift registers until a byte of data is transferred or received.

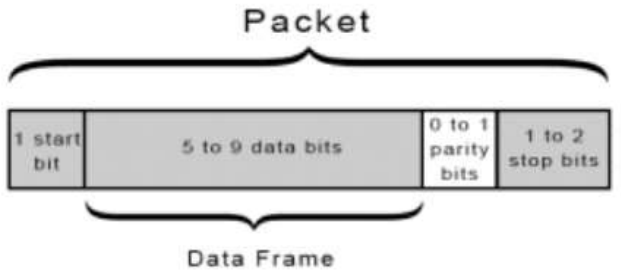
A read or write control logic is also provided to determine when to read or write. The baud rate generator can produce speeds ranging from 110 bps to 230400 bps. For faster data transfer, microcontrollers typically use higher baud rates such as 115200 and 57600. Slower baud rates of 4800 and 9600 are used by devices like GPS and GSM.

4.1.4 HOW UART WORKS

The data for the UART that will transmit it comes from a data bus. Another device, such as a CPU, RAM, or microcontroller, uses the data bus to deliver data to the UART. Data is sent in parallel from the data bus to the transmitting UART. After receiving parallel data from the data bus, the transmitting UART creates the data packet by adding a start bit, a parity bit, and a stop bit. The data packet is then serially output at the Tx pin, bit by bit. The Rx pin on the receiving UART reads the data payload bit by bit. The data is subsequently converted back into parallel form and the start, parity, and stop bits are removed by the receiving UART. Finally, the receiving UART sends the data packet to the data bus on the receiving end in parallel.



The data sent over UART is divided into packets. Each packet has one start bit, five to nine data bits (depending on the UART), an optional parity bit, and one or two stop bits.



Start Bit

When the UART data transmission line is not transmitting data, it is generally held at a high voltage level. The transmitting UART pulls the transmission line from high to low for one clock cycle to initiate data transfer. When the receiving UART detects a high-to-low voltage transition, it starts reading the bits in the data frame at the baud rate's frequency.

Data Frame

The actual data being sent is contained in the data frame. If a parity bit is employed, it can be anything from 5 to 8 bits long. The data frame can be 9 bits long if no parity bit is used. The data is usually delivered with the least significant bit first.

Parity

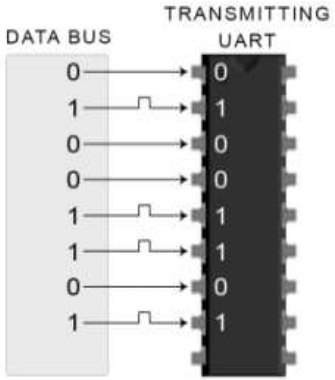
The evenness or oddness of a number is described by parity. The receiving UART uses the parity bit to determine if any data has changed during transmission. Electromagnetic radiation, mismatched baud rates, and long-distance data transmissions can all alter bits. After reading the data frame, the receiving UART counts the number of bits with a value of 1 and determines whether the total is even or odd. The 1 bits in the data frame should amount to an even number if the parity bit is a 0 (even parity). The 1 bits in the data frame should sum to an odd number if the parity bit is a 1 (odd parity). The UART understands that the transmission was error-free when the parity bit matches the data. The UART knows that bits in the data frame have changed if the parity bit is a 0 and the total is odd; or if the parity bit is a 1 and the total is even.

Stop bit

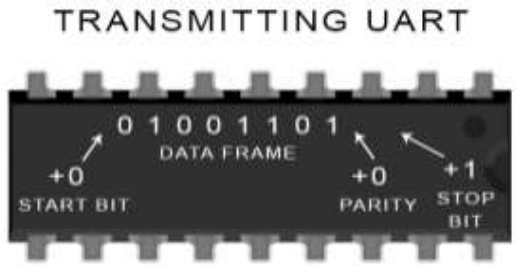
The sending UART drives the data transmission line from a low voltage to a high voltage for at least two bit lengths to signify the end of the data packet.

4.1.5 STEPS OF UART TRANSMISSION

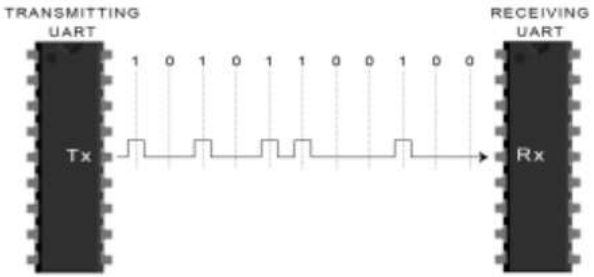
1. The transmitting UART receives data from the data bus in parallel.



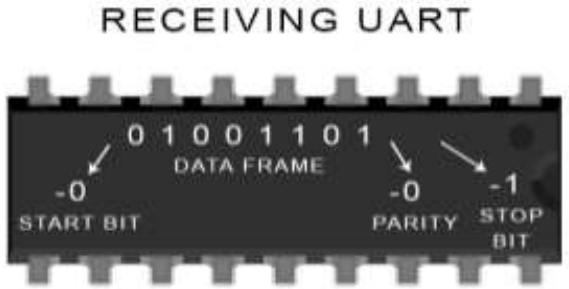
2. The starting bit, parity bit, and stop bit(s) are added to the data frame by the transmitting UART.



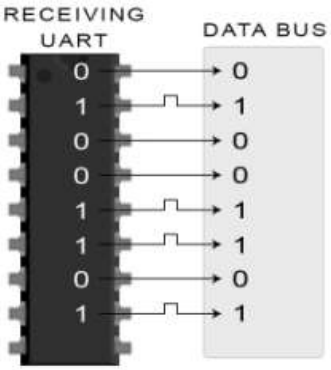
3. From the transmitting UART to the receiving UART, the full packet is transferred serially. The data line is sampled by the receiving UART at the specified baud rate.



4. The data frame's start, parity, and stop bits are discarded by the receiving UART.



5. On the receiving end, the receiving UART translates the serial data to parallel and transfers it on the data bus.



4.1.6 ADVANTAGES OF UART

- Only two wires are used.
- There is no need for a clock signal.
- Has a parity bit that can be used to check for errors.
- The data packet's structure can be modified as long as both sides are prepared.
- This approach is well-documented and commonly used.

4.1.7 DISADVANTAGES OF UART

- The data frame size is restricted to a maximum of 9 bits.
- Multiple slave or master systems are not supported.
- Each UART's baud rates must be within ten percent of one another.

4.2 GPIO

GPIO, or General-Purpose Input Output, is a standard interface for digital input and output found on microcontrollers and SBCs. It enables these devices to control external components such as motors and infrared transmitters (output) as well as receive data from sensor modules and switches (input). In essence, GPIO allows our Raspberry Pi to communicate with a wide range of external components, making it useful for projects ranging from a weather station to a self-driving robot. Software configurations will be necessary for GPIO pins to work. Don't worry; beginner-friendly Python packages like GPIOzero exist to make physical computing more accessible to everyone. GPIO access libraries such as wiringPI are also available for more experienced programmers who prefer C or C++.

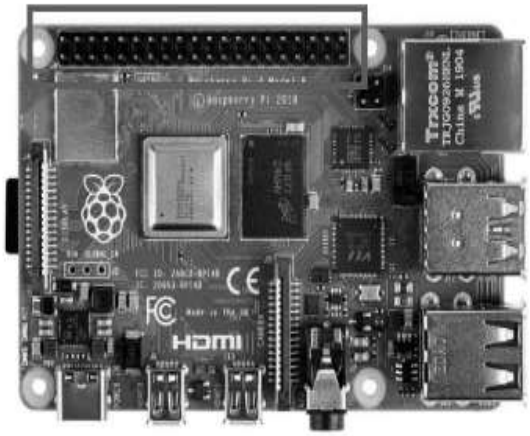


Figure 1.4 Raspberry Pi4 40 Pin GPIO Header

4.2.1 PURPOSE OF THE PERIPHERAL

In order to interact with other components in the system via low-speed interface pins, most devices require some general-purpose input/output (GPIO) functionality. The GPIO peripheral is where you may control and use the GPIO capability on this device.

4.2.2 FEATURES

The following are the characteristics of the GPIO peripheral.

- Separate data set and clear registers provide output set/clear capabilities, allowing several software processes to control GPIO signals without compromising crucial section protection.
- Set/clear functionality is also supported by writing to a single output data register.
- Input/output registers are separated
  - The output register can be read to see the status of the output drive.
  - The input register can be read to see the status of the pins.
- With adjustable edge detection, all GPIO signals can be used as interrupt sources.
- All GPIO signals can be used to send EDMA messages.

4.2.3 FUNCTIONAL BLOCK DIAGRAM

Figure 1.5 below represents the GPIO peripheral block diagram

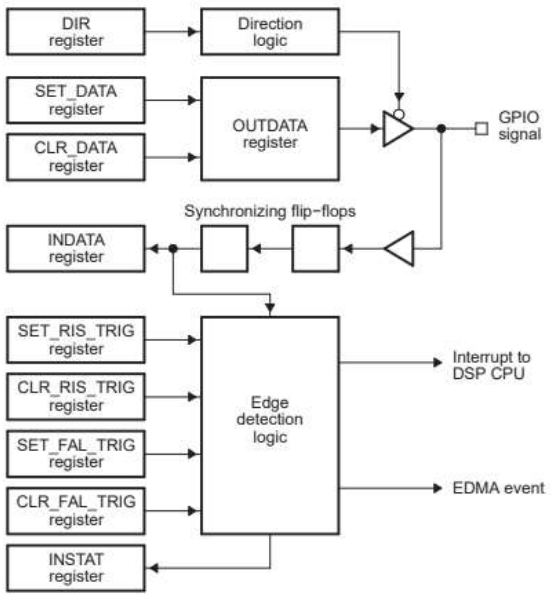


Figure 1.5 GPIO peripheral block diagram

4.2.4 RASPBERRY PI GPIO PINOUT

On the GPIO header of the Raspberry Pi B+, 2, 3, Zero, or the latest Raspberry Pi 4 Model B, you'll find a total of 40 GPIO pins. Older RPI models, such as the Raspberry Pi Model B, only have 26 pins.

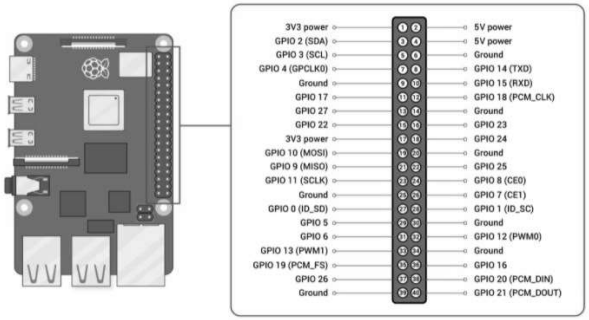


Figure 1.6 Raspberry Pi4 GPIO pin header

Each pin on the 40-pin header has a specific purpose. In the table below, the various categories are described.

GPIO pin type	Pin functionality
GPIO	GPIO pins are general – purpose pins that can be used to switch external devices on and off, such as an LED.
Power	External components are supplied with 5V and 4.3V po wer via the 5V and 3V3 pins.
I2C	I2C pins are used to connect and communicate with external modules that are I2C compliant.
SPI	Hardware communication is also done via SPI (Serial Peripheral Interface Bus) pins, although with a different protocol.
UART	For serial communication, UART (Universal Asynchronous Receiver/Transmitter) pins are utilized.
DNC	DNC (Do Not Connect) pins should be avoided at all costs.
GND	GND (Ground) pins are pins in your circuits that offer electrical grounding.

4.2.5 CONFIGURING GPIO PIN

You can skip these steps and get right into programming with GPIO if you're using the latest version of Raspberry Pi OS.

Otherwise, you'll have to update your RPI with the following commands in the serial terminal:

sudo apt-get update

sudo apt-get upgrade

If you don't have the GPIO package loaded for some reason, execute the following command to install it:

sudo apt-get install rpi.gpio

4.2.6 ESSENTIAL PRODUCTS FOR RASPBERRY PI GPIO

- Grove Base Hat for Raspberry Pi

  The Grove Base Hat adds 15 Grove connectors to the Raspberry Pi's initial 40 GPIO pins, extending the device's capabilities. Grove is a modular, standardized connector system that eliminates the need for jumper wires or solder for rapid and easy electronics prototyping.

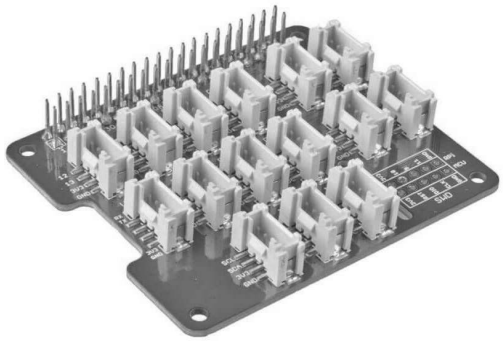
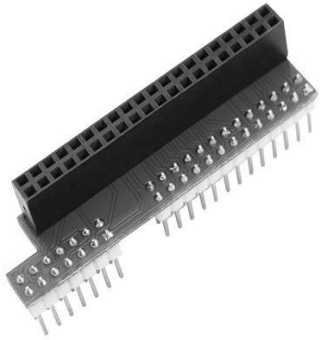


Figure 1.7 Grove Base Hat for Raspberry Pi

The Grove Base Hat allows the Raspberry Pi to connect to the Grove ecosystem, which includes over 300 sensors, actuators, and communication modules. Getting started with Raspberry Pi GPIO projects has never been easier than it is now, thanks to Grove's libraries and clear documentation.

- Raspberry Pi 40pin to 26pin GPIO Board

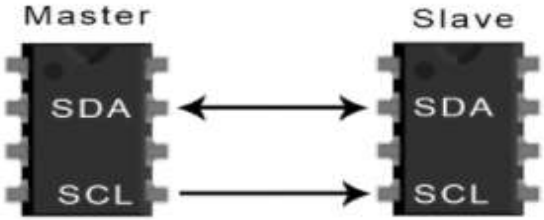
  This 40-Pin to 26-Pin GPIO adapter board will come in handy if you have older Raspberry Pi accessories that were designed for the original 26-Pin layout. This GPIO board transforms the latest Raspberry Pi models' 40-pin header to the original 26-pin layout, allowing you to use your existing Raspberry Pi accessories.



4.3 I2C

I2C brings together the greatest aspects of SPI and UARTs. Numerous slaves can be connected to a single master (like SPI) via I2C, and multiple masters can control single or multiple slaves. When you wish to have multiple microcontrollers logging data to a single memory card or displaying text on a single LCD, this is really beneficial.

I2C employs only two wires to send data between devices, similar to UART communication.



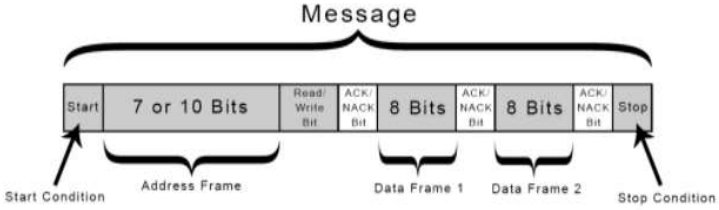
- SDA (Serial Data): The data transmission and reception line between the master and slave.
- Serial Clock Line (SCL): This is the line that carries the clock signal.

I2C is a serial communication technology, which means data is sent bit by bit over a single wire (the SDA line).

I2C, like SPI, is synchronous, which means that a clock signal shared by the master and slave synchronizes the output of bits with the sampling of bits. The master is always in charge of the clock signal.

4.3.1 WORKING OF I2C

I2C sends data in the form of messages. Frames of data are used to break up messages. Each message consists of an address frame with the slave's binary address and one or more data frames containing the data to be delivered. Between each data frame, the message additionally comprises start and stop conditions, read/write bits, and ACK/NACK bits:



- Start Condition: Before the SCL line shifts from high to low voltage, the SDA line switches from high to low voltage.
- Stop Condition: After the SCL line switches from low to high voltage, the SDA line switches from low to high voltage.
- Address frame: When the master wants to talk to a slave, it uses an address frame, which is a 7- or 10-bit sequence that uniquely identifies the slave.

- Read/Write bit: A single bit indicating whether the master is providing data to the slave (low voltage level) or requesting data from it (high voltage level) (high voltage level).

- ACK/NACK Bit: An acknowledge/no-acknowledge bit follows each frame in a communication. The receiving device returns an ACK bit to the sender if an address frame or data frame was successfully received.

- **Addressing**

  Because I2C lacks slave select lines like SPI, it requires a different method of informing the slave that data is being transmitted to it and not to another slave. It accomplishes this through addressing. In a new message, the address frame is always the first frame after the start bit.

  Every slave connected to the master receives the address of the slave with whom it wishes to interact. After that, each slave compares the address sent by the master to its own. It sends a low voltage ACK signal back to the master if the addresses match. The slave does nothing if the addresses do not match, and the SDA line remains high.

- **Read/Write bit**

  A single bit at the end of the address frame tells the slave whether the master wishes to write data to it or receive data from it. The read/write bit is a low voltage level if the master wants to send data to the slave. The bit is a high voltage level if the master is seeking data from the slave.

- **Data Frame**

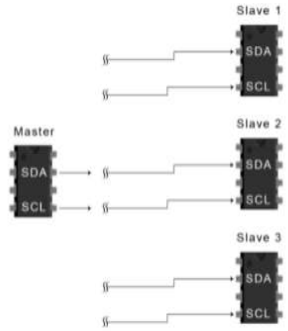
  The initial data frame is ready to be delivered after the master detects the ACK signal from the slave.

  The data frame is always 8 bits long, and the most significant bit is always sent first. Each data frame is immediately followed by an ACK/NACK bit to confirm that it was successfully received. Before the next data frame can be delivered, the ACK bit must be received by either the master or the slave (depending on who is transmitting the data).

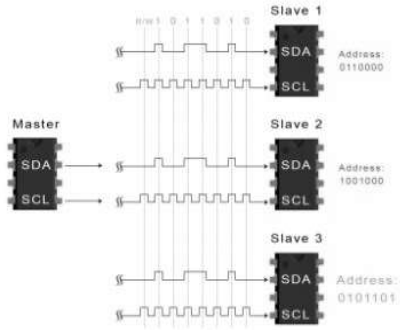
  The master can send a stop condition to the slave to interrupt the transmission when all of the data frames have been sent. After a low to high transition on the SCL line, the stop condition is a voltage transfer from low to high on the SDA line, with the SCL line remaining high.

4.3.2 I2C DATA TRANSMISSION STEPS

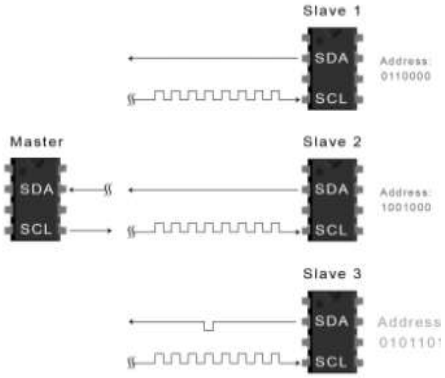
1. Before moving the SCL line from high to low, the master sends the start condition to all linked slaves by switching the SDA line from high to low voltage:



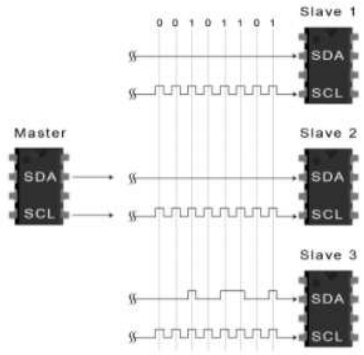
2. The master sends the read/write bit and the 7 or 10 bit address of the slave it wants to connect with to each slave:



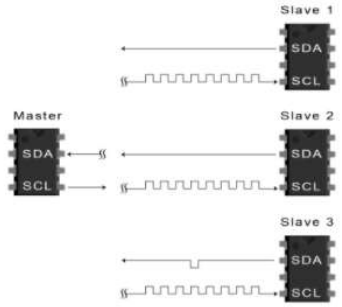
3. Each slave checks the address sent by the master against its own. The slave returns an ACK signal by pulling the SDA line low for one bit if the addresses match. The slave leaves the SDA line high if the master's address does not match the slave's own address.



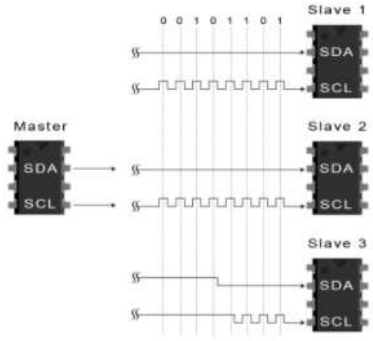
4. The data frame is sent or received by the master:



5. The receiving device sends another ACK bit to the sender after each data frame has been delivered to acknowledge successful reception of the frame:

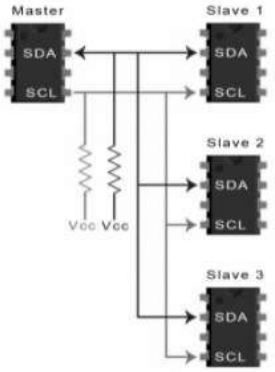


6. The master transmits a stop condition to the slave by switching SCL high before switching SDA high to cease data transmission:



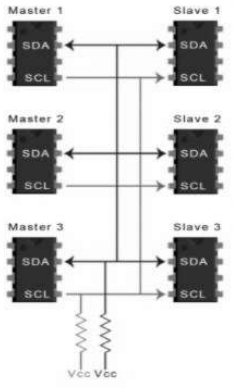
4.3.3 SINGLE MASTER MULTIPLE SLAVES

Because I2C employs addressing, a single master can control numerous slaves. There are 128 (27) unique addresses possible using a 7 bit address. It's unusual to use 10 bit addresses, yet they provide 1,024 (210) unique addresses. If you want to link numerous slaves to a single master, use 4.7K Ohm pull-up resistors to connect the SDA and SCL lines to Vcc.



4.3.4 MULTIPLE MASTER MULTIPLE SLAVES

A single slave or several slaves can be tied to multiple masters. When two masters in the same system try to send or receive data over the SDA line at the same time, the problem arises. To overcome this issue, each master must first determine if the SDA line is low or high before sending a message. If the SDA line is low, another master is in charge of the bus, and the master should hold off on sending the message. It is safe to transfer the message if the SDA line is high. Use the following schematic, with 4.7K Ohm pull-up resistors connecting the SDA and SCL lines to Vcc, to connect many masters to multiple slaves.



4.3.5 ADVANTAGES

- Only two wires are used.
- Multiple masters and slaves are supported.
- The ACK/NACK bit indicates whether each frame was successfully transferred.
- The hardware is simpler than using UARTs.
- Protocol that is well-known and extensively utilized

4.3.6 DISADVANTAGES

- Data transport rate is slower than SPI.
- The data frame size is limited to 8 bits.
- Hardware that is more difficult to implement than SPI is required.

4.4 SPI

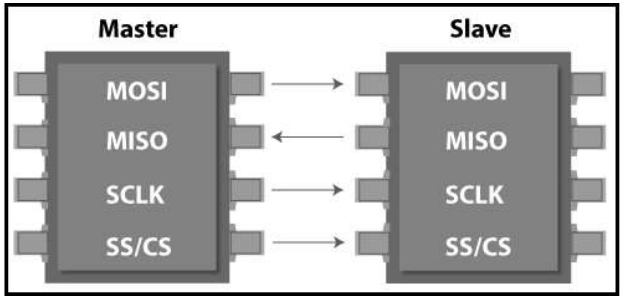
Serial Peripheral Interface (SPI) stands for Serial Peripheral Interface. It's a serial communication protocol used to link low-speed devices together. Motorola created it in the mid-1980s for inter-chip communication. It's frequently used to communicate with flash memory, sensors, real-time clocks (RTCs), and analog-to-digital converters, among other things. It's a full-duplex synchronous serial communication, which means data can be sent in both directions at the same time.

The fundamental benefit of the SPI is that it allows data to be transferred without interruption. This protocol allows for a large number of bits to be broadcast or received at once.

Devices communicate using this protocol in a master-slave relationship. The slave device is controlled by the master device, and the slave device follows the master device's instructions. A single slave and a single master is the most basic arrangement of the Serial Peripheral Interface (SPI). One master device, on the other hand, can control several slave devices.

4.4.1 SPI INTERFACE

The communication in the SPI protocol is done via four wires. They are depicted in the diagram.



- MOSI: MOSI (Master Output Slave Input) is an acronym that stands for Master Output Slave Input. It's utilized to transfer data between the master and the slave.
- MISO: MISO (Master Input Slave Output): MISO stands for Master Input Slave Output. It's utilized to transfer data between the slave and the master.
- SCL/SCLK: The clock signal is denoted by the letters SCK or SCLK (Serial Clock).
- SS/CS: The master uses SS/CS (Slave Select / Chip Select) to deliver data by selecting a slave.

NOTE: If only one slave is present in the communication, only three wires are necessary. It does not require the SS (slave select).

4.4.2 CHARACTERISTICS OF SPI BUS

- The maximum frequency has yet to be determined. The bus can travel as quickly as your chips and board design allow
- Data transmissions of 25-50 Mbits/sec are possible
- The Serial Data
- Point-to-Point topology is simple to implement and allows transceivers to convert SPI signaling to RS485, CAN, fiber-optic, and other protocols. The SPI protocol is unaffected, thus long-distance and isolated connections are possible.

4.4.3 MULTI-DEVICE TOPOLOGIES

The daisy-chain and star multi-device topologies are supported by SPI. The clock is split in two by the Daisy-chain topology, allowing it to route in parallel to the slaves. However, data is still point-to-point. The MISO of one slave is linked to the MOSI of another, forming a chain. Similar to a boundary scan, data for all devices clocks through all devices in a chain; each device just selects out the data intended to it. The chain's final device sends its MISO to the master.

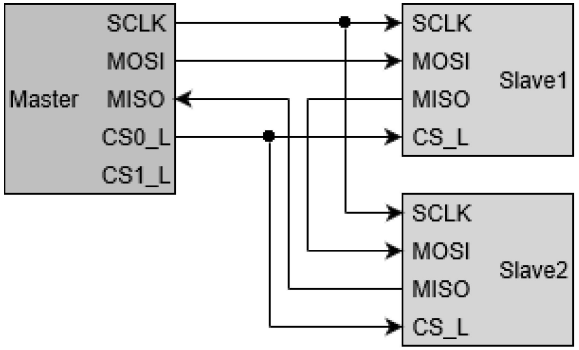


Figure 1.8 SPI Bus – Daisy Chain Topology

Except for chip select, all signals in the Star topology are separated and routed to each slave in parallel. Individual slave devices are selected using multiple chip select. This mode is supported by more devices than daisy-chaining.

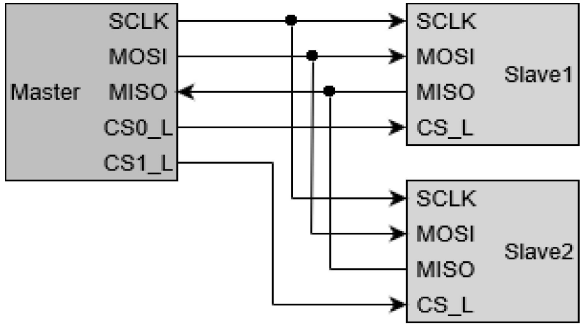


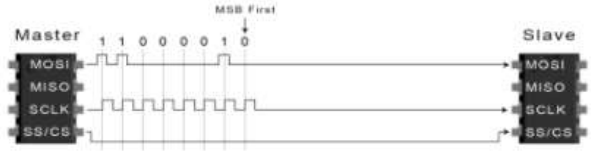
Figure 1.9 SPI Bus – Star Topology

4.4.4 SPI DATA TRANSMISSION STEPS

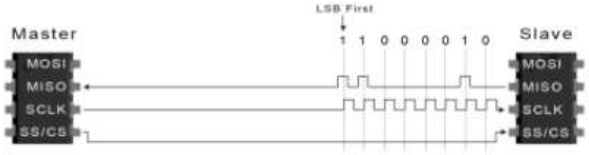
1. The clock signal is output by the master.

2. The master activates the slave by lowering the voltage on the SS/CS pin.

3. Along the MOSI line, the master transfers the data one bit at a time to the slave. As the bits are received, the slave reads them.



4. If a response is required, the slave sends data to the master one bit at a time via the MISO line. As the bits are received, the master reads them.



4.4.5 ADVANTAGES

- The fundamental benefit of the SPI is that it allows data to be transferred without interruption.
- Simple in hardware.
- It can communicate in full duplex mode.
- In this protocol, the slave does not require a unique address.
- Because it uses the master's clock, this protocol does not necessitate accurate slave device oscillation.
- The software implementation is straightforward in this case.
- It has a fast transfer rate.
- Signals are only sent in one direction.
- It contains independent MISO and MOSI lines, allowing data to be delivered and received simultaneously.

4.4.6 DISADVANTAGES

- It usually only supports one master.
- Unlike the UART, it does not check for errors.
- It has a larger number of pins than the other protocol.
- Only from a limited distance can it be used.
- It makes no acknowledgement of whether or not the data has been received.

4.4.7 APPLICATIONS

- Memory: SD Card, MMC, EEPROM, and Flash memory are all options.
- Sensors: Temperature and pressure sensors are used.
- Control devices: ADC, DAC, digital POTS, and Audio Codec are the control devices.
- Others: Other features include a camera lens mount, a touchscreen, an LCD, an RTC, a video game controller, and so on.

4.5 SUMMARY

Different Raspberry Pi interfaces such as UART, GPIO, I2C, SPI is explored. The UART interface of Raspberry Pi is used for serial communication. General purpose I/O is also investigated. For example, GPIO 14 can be an input, an output, or a serial port TX data line. As a result, the Raspberry Pi is extremely adaptable. The Pi's GPIO interface has a weak CMOS 3 V interface, which is one of the issues. The I/O pins are weak drivers, and the GPIO pins are prone to static electricity harm (2 to 16 mA). GPIO power must also be budgeted from the 50 mA total spare current capacity. Using adapter boards solves these issues however it comes at a high price. This creates a fertile ground for developing low-cost, high-effective roll-your-own solutions. The concept of I2C bus is also explored. Philips invented the I2C bus, commonly known as the two-wire interface (TWI), in 1982 to facilitate communication with slower devices. It was also cost-effective because it just required two wires (excluding ground and power). Other standards, such as the SMBus, have been developed since then, expanding on this structure. The original I2C bus, on the other hand, continues to be popular as a simple and cost-effective means to connect peripherals. Followed by this the SPI technique is also discussed. The Serial Peripheral Interface bus, or spy for short, is a synchronous serial interface created by Motorola. The SPI protocol works in full-duplex mode, which means it may send and receive data at the same time. In general, SPI outperforms the I2C protocol in terms of speed, but it necessitates more connections. Lastly the useful implementation such as cross compilation technique, pulse width modulation and the interface for camera has been studied.

4.6 LIST OF REFERENCES

1) Mastering the Raspberry Pi, Warren Gay, Apress(2014)
2) https://mitu.co.in/wp-content/uploads/2017/09/03-Raspbian-Operating-System.pdf
3) https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners
4) https://www.geeksforgeeks.org/linux-commands/
5) https://www.javatpoint.com/nodejs-tutorial
6) https://www.tutorialspoint.com/nodejs/index.htm
7) https://www.w3schools.com/python/
8) https://docs.python.org/3/tutorial/
9) https://www.javatpoint.com/python-tutorial
10) https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html

11) https://www.ti.com/lit/ug/spruf95/spruf95.pdf?ts=1632378909735&ref_url=https%253A%252F%252Fwww.google.com%252F
12) https://www.seeedstudio.com/blog/2020/02/19/how-to-use-raspberry-pi-gpio-pins-python-tutorial/
13) https://www.ti.com/lit/an/slva704/slva704.pdf?ts=1632361805005&ref_url=https%253A%252F%252Fwww.google.com%252F
14) https://embetronicx.com/tutorials/tech_devices/i2c_1/
15) https://practicalee.com/spi/
16) http://events17.linuxfoundation.org/sites/events/files/slides/Shuah_Khan_cross_compile_linux.pdf
17) https://projects.raspberrypi.org/en/projects/getting-started-with-picamera

4.7 UNIT END EXERCISES

1] Write a note on UART.
2] Explain the block diagram of UART and explain in brief why it is used?
3] State the advantages and disadvantages of UART.
4] Write a note on UART transmission steps.
5] Discuss the purpose, features of GPIO.
6] Write a note on working of I2C and state its advantages and disadvantages.
7] Describe the various I2C data transmission steps.
8] Write a note on single master multiple slaves.
9] Explain the concept of multiple master multiple slaves.
10] Write a note on SPI interfaces along with its characteristics.
11] Describe the SPI multidevice topologies.
12] Write a note on advantages, disadvantages and applications of SPI.
13] Explain the SPI data transmission steps.

❄❄❄❄❄❄❄

5

USEFUL IMPLEMENTATIONS

Unit Structure

5.0 OBJECTIVES

After going through this unit, you will be able to:

- Understand the fundamentals of cross compilation
- Acquaint with the concepts of pulse width modulation
- Interfacing of SPI for camera with its applications

5.1 CROSS COMPILATION

A compiler is a piece of software that converts source code to executable code. A compiler, like all programs, operates on a specific type of computer, and the new programs it generates run on the same type of computer.

The computer on which the compiler runs is known as the host, whereas the computer on which new programs execute is known as the target. The compiler is a native compiler when the host and target machines are of the same type. The compiler is a cross compiler when the host and target are different. The act of compiling code for one computer system (commonly referred to as the target) on a different computer system (often referred to as the host) is known as cross-compilation. When the target system is too small to host the compiler and all essential files, this is a highly handy strategy.

- **Where does cross compiler come into play?**

  Cross compiler is used in Bootstrapping. Meaning- Getting started on a new platform. A cross compiler is used to compile necessary tools such as the OS and a native compiler when developing software for a new platform.

5.1.1 NEED OF CROSS COMPILERS

In theory, a PC user might get the proper target hardware (or emulator), boot a Linux distro on it, and compile natively within that environment. While this is a valid strategy (and perhaps even a good one when dealing with a Mac Mini), it has a few significant drawbacks when dealing with items like a Linksys router or an iPod.

- **Speed -** Target platforms are often an order of magnitude or slower than hosts. The majority of special-purpose embedded hardware is made for low cost and low power consumption, rather than for high performance. By virtue of running on high-powered desktop hardware, modern emulators (like qemu) are actually quicker than a lot of the real-world hardware they simulate.
- **Capability -** Compiling consumes a lot of resources. The target platform typically lacks the resources of a desktop, such as gigabytes of memory and hundreds of gigabytes of disc space; it may not even have the resources to generate "hello world," let alone huge and complex packages.
- **Availability -** A cross-compiler is required to bring Linux up on a hardware platform it has never run on before. Finding an up-to-date full-featured prebuilt native environment for a given target, even on long-established platforms like Arm or Mips, can be difficult. If the platform isn't typically used as a development workstation, there may not be a recent prebuilt distro available, and if there is, it's likely out of date. You're back to cross-compiling anyway if you have to build your own distro for the target before you can build on the target.
- **Flexibility -** A fully functional Linux distribution has hundreds of packages, but in most cases, a cross-compile environment can rely on the host's existing distro. Cross compiling focuses on constructing the target packages to be deployed rather than spending time on the target system for build-only prerequisites to work.
- **Convenience -** The user interface of headless boxes can be a little claustrophobic. It's difficult enough to diagnose build errors as it is. It's a hassle to install software from a CD onto a machine that doesn't have a CD-ROM drive. It's wonderful to be able to recover from accidently lobotomizing your test system rather of having to reboot back and forth between your test environment and your development environment.

5.1.2 WHY CROSS COMPILING IS DIFFICULT?

- **Portable native compiling is hard.**

  It's difficult to compile native code in a portable format. The majority of applications are written on x86 hardware and compiled natively. Cross-compiling thus encounters two categories of issues: issues with the applications themselves and issues with the build mechanism.

  The first sort of issue affects all non-x86 targets, both native and cross-built versions. Most programs make assumptions about the sort of system they operate on, and these assumptions must match the platform in issue or the program will not run. The following are some common assumptions:

- **Word size -** On a 64-bit platform, copying a pointer into an int may lose data, and calculating the size of a malloc by multiplying by 4 instead of sizeof(long) isn't ideal. Integer overflow issues can sometimes be subtle, such as "if (x+y size) memset(src+x,0,y);", which results in a 4 GB memset on 32-bit hardware when x=1000 and y=0xFFFFFFF0...
- **Endianness -** Different systems store binary data internally in different ways, requiring translation when reading int or float data from disc or the network.
- **Alignment -** Some platforms (such as arm) can only read or write integers from addresses that are an even multiple of four bytes, or they may segfault. Even those that can tolerate arbitrary alignments are slower when dealing with unaligned data (they must fetch both halves twice), hence the compiler will frequently pad structures to align variables. Treating structures as a blob of data that can be written to disc or delivered over the network necessitates additional effort to assure consistency.
- **Default signedness -** Whether the "char" data type is signed or unsigned by default varies from platform to platform (and, in some situations, from compiler to compiler), which might result in some unexpected issues. The simple solution is to use a compiler parameter such as "-funsigned-char" to force the default value to a known value.
- **NOMMU -** If your target platform lacks a memory management unit, you'll need to make a few adjustments. Only certain sorts of mmap() work (shared or read only, but not copy on write), and the stack doesn't grow dynamically, so you'll need vfork() instead of fork().

Most packages seek to be portable when compiled natively, and will at the very least accept patches given to the proper development mailing list to remedy any of the above concerns (with the possible exception of NOMMU difficulties).

- **Cross- compiling**

  Cross-compiling has its own set of challenges in addition to native compiling's:

- **Configuration difficulties -** To be portable when natively compiled, packages having a separate configuration step (the "./configure" section of the typical configure/make/make install) frequently test for factors like endianness or page size. Because these values differ across the host and target systems when cross-compiling, performing tests on the host system yields incorrect results. When the target doesn't have that package or has an incompatible version, configuration can detect its presence on the host and include support for it.
- **HOSTCC vs. TARGETCC -** Many build procedures, such as the above configuration tests, or programs that generate code (such as a C program that generates a.h file that is then #included during the main build), need compiling items to execute on the host system. Simply substituting a target compiler for the host compiler damages packages that require the build of objects that run during the build process. These packages require access to both a host and a target compiler, as well as instruction on when to use each.
- **Toolchain Leaks -** An incorrectly configured cross-compile toolchain can leak pieces of the host system into built applications, causing failures that are normally easy to detect but complex to diagnose and fix. At link time, the toolchain may #include the incorrect header files or search the incorrect library directories. Shared libraries frequently rely on other shared libraries, which can introduce unintended host-system link-time references.
- **Libraries -** At compile time, dynamically linked applications must access the proper shared libraries. In order for programs to link against shared libraries on the target system, they must be included to the cross-compile toolchain.
- **Testing -** The development system provides a handy testing environment for native builds. Confirming that "hello world" compiled properly while cross-compiling can necessitate configuring (at the very least) a bootloader, kernel, root file system, and shared libraries.

5.1.3 WORKING OF CROSS COMPILATION

A cross compiler is a compiler that can generate executable code for platforms other than the one on which it is currently operating. In paravirtualization, a single machine runs numerous operating systems, and a cross compiler might build executable for each from a single source. The ultimate purpose of several separate components is to produce the byte code that the target CPU utilizes. You've successfully cross-compiled when you can generate the assembled byte code. Any compiler's key components are:

- **Parser:** The parser translates the source code of the raw language to assembly language. The parser must be familiar with the destination assembly language because you're converting from one format to another (C to assembly).

- **Assembler:** The assembler translates assembly language code into byte code, which is then executed by the CPU.

- **Linker:** The linker assembles the individual object files generated by the assembler into a single executable application. Encapsulation mechanisms and standards vary depending on the operating system and CPU mix. To function, the linker must be aware of the target format.

- **Standard C library:** A central C library contains the essential C functions (for example, printf). If the application uses functions from the C library, this library is utilized in conjunction with the linker and the source code to build the final executable.

  Each of these components of a standard host-based C compiler is designed to produce the host's associated assembly code, byte code, and target execution format. Although the application is meant to run on the host, the assembly language, linker, and C library are all created for the target platform and processor in a cross-compiler. You might cross-compile an application on an Intel-based Linux computer so that the assembly language and final application are for a Solaris-based SPARC host.

  As a result, creating a cross-compiler necessitates creating a different version of the C compiler suite that creates and links applications for the target host. You can develop your own cross-compilers since you can compile GCC and the related tools.

5.1.4 BUILD PROCESS OF CROSS COMPILER

The GNU utilities (that is, the GCC), which include the C compiler, binary utilities, and the C library, have a number of advantages, the most notable of which is that they are free, open source, and simple to compile. From a cross-compiler standpoint, the fact that GCC has been ported to a variety of systems means that the code supports a variety of CPU and platform types. However, there are certain limits. GCC does not support all processor kinds or systems (albeit it does produce the majority). When you run the configuration tools, you'll get a warning.

You'll need three components from the GNU suite to make a cross-compiler:

- **binutils:** Basic binary utilities like the assembler and linker, as well as related tools like Size and Strip, are included in the binutils package. Both the essential components for generating an application and the tools that may be used to build and edit the target execution format are included in the binary utilities. The Strip utility, for example, eliminates symbol tables, debugging, and other "useless" information from an object file or application, but it has to know the target format to avoid removing the erroneous data.

- **gcc:** The major component of the compilation process is the gcc. Gcc is made up of two parts: a C preprocessor (cpp) and a translator that transforms C code to the target CPU assembly language. Gcc also serves as a user interface for the entire process, invoking cpp, the translator, the assembler, and the linker as needed.

- **newlib/glibc:** This library is the standard C library. Newlib was created by Redhat and may be slightly more user-friendly in cross-compilers intended for embedded targets.

  You'll also need the target operating system's header files, which are required so that you can access all of the operating system's functions and system calls needed to build the program. The headers are relatively easy to obtain on Linux. You can copy an existing set of headers for various operating systems.

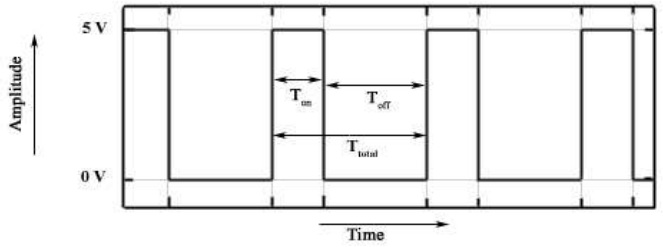
  You can also construct the GNU debugger - gdb - for the target host if you like. Because emulation is required, you can't create a debugger that can execute code for the target while running on the host. You can, however, create a gdb executable for your target host.

5.2 PULSE WIDTH MODULATION

PWM (Pulse Width Modulation) is a technique for varying the width of pulses in a pulse train. It's a digital technique that manipulates the quantity of power given to a gadget. It uses a digital source to generate analogue signals. A PWM signal is a square wave that alternates between on and off states. A PWM signal's behavior is determined by its duty cycle and frequency. PWM is used to operate servos and speed controllers, as well as limit the effective power of motors and LEDs.

5.2.1 PWM principle

A square wave with changing high and low times is what pulse width modulation is. The following diagram depicts a basic PWM signal.



Several terms are associated with PWM such as

- **ON Time:** The duration of the time signal when it is ON is high.

- **OFF Time:** The duration of the time signal is low.

- **Period:** The sum of the on-time and off-time of a PWM signal is the period.
- **Duty cycle:** The percentage of time that the signal remains on during the period of the PWM signal is referred to as duty cycle.
- **Frequency:** The time it takes for this signal to complete a one-and-off cycle is measured in periods. The frequency is the inverse of the period, and it is the number of times a periodic change is accomplished per unit time. It establishes the rate at which the PWM completes one cycle, i.e., the rate at which the signal flips from high to low states. The output will behave like an analogue signal with a constant voltage if we turn the digital signal on and off with a high enough frequency.
- **Period:**

  Ton signifies the signal's on-time, and Toff denotes the signal's off-time, as illustrated in the diagram. Period is determined as the sum of both on and off times, as stated in the equation below.

$$T_{Total} = T_{ON} + T_{OFF}$$

- **Duty cycle:**

  The on-time of the period of time is used to determine the duty cycle. Using the above-mentioned period, the duty cycle is determined as follows:

$$D = \frac{T_{on}}{T_{on}+T_{off}} = \frac{T_{on}}{T_{total}}$$

5.2.2 APPLICATIONS OF PWM

- Adjusting screen brightness: PWM can be used to adjust the brightness of the screen. Adjusting the brightness of the screen via PWM does not rely on electricity, but rather on the screen alternating on and off. When the PWM dimming screen is turned on, it does not output light continuously, but it does light up and switch off the screen frequently. If this changes quickly enough, our eyes will perceive it as always on, but with varying brightness dependent on duty cycles. The brighter the screen, the higher the duty cycle.
- Set the volume of the buzzer to a different level.
- Control the motor's speed.
- A servo's direction can be controlled.
- Providing analog output.
- Create an audio signal
- Telecommunication: Message encoding

5.3 SPI FOR CAMERA

Since 2012, the Arducam team has been developing the world's first high-resolution SPI camera solution for Arduino, which fills a gap in the Arduino community's camera supply. These SPI cameras are general-purpose solutions that can be used on any hardware platform that has the SPI and I2C interfaces, not just the Arduino platform. The SPI bus' flexibility increases the utility of the SPI camera by allowing customers to connect several cameras to a single microcontroller and shoot images at the same time. Support for LCD screens is optional.

Universal SPI Camera Shield	SPI Mini Camera Shield
It hides the complex nature of the camera and provides the plug and play camera control interface as well as the ready to use software source code library and demo code	It can be used in many platforms like Arduino, Raspberry Pi, Maple, Chipkit, Beaglebone black, as long as they have SPI and I2C interface and can be well mated with standard Arduino boards

The Raspberry Pi Pico, as an alternative to Arduino, lacks processing power, memory, and a CSI interface, making it incompatible with the official or any MIPI CSI-2 camera modules. Pico, fortunately, has a variety of versatile I/O choices, including SPI, which allows the Arducam SPI camera to function with Pico.

5.3.1 APPLICATIONS

- Cameras for internet of things (IoT) applications
- Cameras for robots
- Camcorders for wildlife
- Other battery-operated devices
- MCU, Raspberry Pi, ARM, DSP, and FPGA platforms are all compatible.

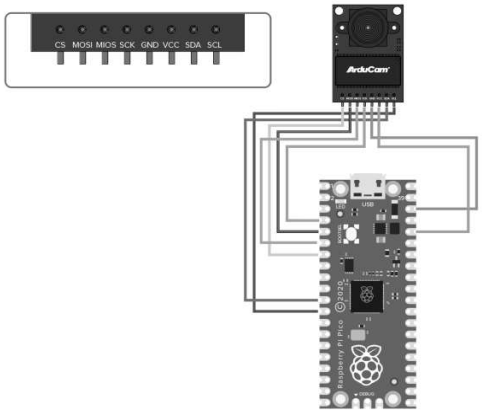
5.3.2 FEATURES

- OV2640 (B0067) 2MP image sensor / OV5642 5MP image sensor (B0068)
- Lens holder for M12 or CS mounts with interchangeable lenses
- With the right lens combination, IR sensitive
- Sensor setup through the 12C interface
- Camera commands and data stream are sent over the SPI interface.
- All of the IO ports are 5V/3.3V compatible.
- JPEG compression mode, single and multiple shoot mode, one-time capture multiple read operation, burst read operation, low power mode, and other features are all supported.
- Standard Raspberry Pi Pico boards are well matched.
- Open-source code libraries for Arduino, STM32, Chipkit, Raspberry Pi, and BeagleBone Black are available.
- Small form factor

5.3.3 PIN DEFINITION

Pin No.	Pin Name	Type	Description
1	CS	Input	SPI slave chip select input
2	MOSI	Input	SPI master output slave input
3	MISO	Output	SPI master input slave output
4	SCLK	Input	SPI serial clock
5	GND	Ground	Power ground
6	+5V	POWER	5V Power supply
7	SDA	Bi-directional	Two-Wire Serial Interface Data I/O
8	SCL	Input	Two-Wire Serial Interface Clock

5.3.4 WIRING



Connect SPI Camera to Pico

Camera	CS	MOSI	MISO	SCK	GND	VCC	SDA	SCL
Pico	GP5	GP3	GP4	GP2	GND	3V3	GP8	GP9

5.4 SUMMARY

This unit made us familiar with the fundamentals of SPI technique. The Serial Peripheral Interface bus, or spy for short, is a synchronous serial interface created by Motorola. The SPI protocol works in full-duplex mode, which means it may send and receive data at the same time. In general, SPI outperforms the I2C protocol in terms of speed, but it necessitates more connections. Lastly the useful implementation such as cross compilation technique, pulse width modulation and the interface for camera has been studied.