NEW SYLLABUS

SECOND YEAR B. Sc.
COMPUTER SCIENCE

SEMESTER-IV

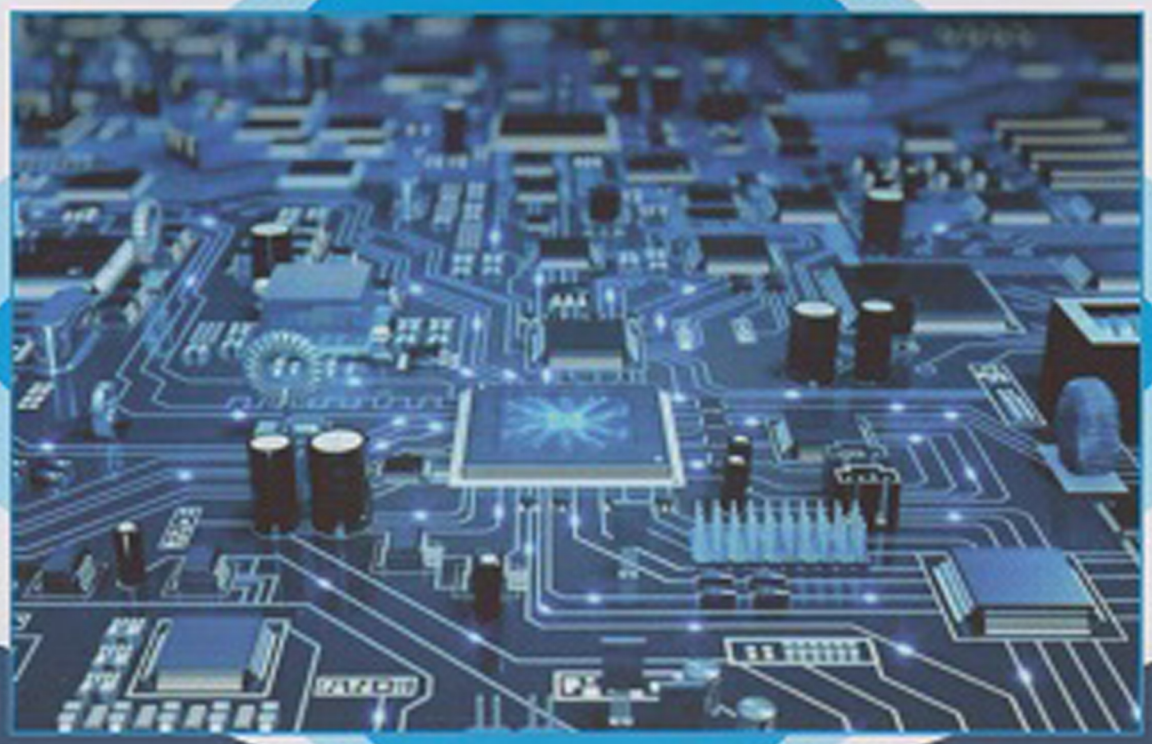
EMBEDDED SYSTEM DESIGN

Electronics (ELC-241) : Paper-I

Dr. J. A. BANGALI
Prof. (Dr.) M. L. DONGARE

S. R. CHAUDHARI,
Prof. (Dr.) P. B. BUCHADE

CBCS
2 CREDITS



NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

SPPU New Syllabus

EMBEDDED SYSTEM DESIGN

[ELC - 241]
For
S.Y.B.Sc. (Computer Science) Semester - IV
Electronics - (Paper - I)
New Syllabus as per CBCS Pattern Credit - 2
June 2020

Dr. J. A. Bangali
M. Sc., M. Phil, Ph.D.
Head, Dept. of Electronic Science,
Co-ordinator, B.Sc. Computer Science
Kaveri College of Arts, Science & Commerce
Pune-411038

S. R. Chaudhari
M. Sc., M.Phil.
Vice principal and Head,
Dept. of Electronic Science,
Modern College, Shivajinagar,
Pune – 411005.

Prof. (Dr.) M. L. Dongare
M.Sc., D.H.E., Ph.D.
Vice Principal, Head, Dept. of Electronic Science,
Rayat Shikshan Sanstha's,
S. M. Joshi College,
Hadapsar, Pune – 411028.

Prof. (Dr.) P. B. Buchade
M.Sc., M.Phil., Ph.D.
Principal and Head,
Dept. of Electronic Science,
Abasaheb Garware College,
Karve Road, Pune - 411004.

Price ₹ 140.00

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

N5546

Embedded System Design **ISBN 978-93-90506-47-7**

First Edition : January 2021

Published By :
NIRALI PRAKASHAN
Abhyudaya Pragati, 1312, Shivaji Nagar,
Off J.M. Road, PUNE – 411005
Tel - (020) 25512336/37/39, Fax - (020) 25511379
Email : niralipune@pragationline.com

Polyplate

Printed By :
YOGIRAJ PRINTERS AND BINDERS
Works: Sr. No. 10\1,Ghule Industrial Estate,
Nanded Village Road,
TAL-HAVELI, DIT-PUNE 411041.
Mobile - 9850046517, 9404225254

☞ DISTRIBUTION CENTRES

PUNE

Nirali Prakashan (For orders within Pune) : 119, Budhwar Peth, Jogeshwari Mandir Lane, Pune 411002, Maharashtra
Tel : (020) 2445 2044, 66022708, Fax : (020) 2445 1538; Mobile : 9657703145
Email : bookorder@pragationline.com, niralilocal@pragationline.com

Nirali Prakashan (For orders outside Pune) : S. No. 28/27, Dhyari, Near Pari Company, Pune 411041
Tel : (020) 24690204 Fax : (020) 24690316; Mobile : 9657703143
Email : dhyari@pragationline.com, bookorder@pragationline.com

MUMBAI

Nirali Prakashan : 385, S.V.P. Road, Rasdhara Co-op. Hsg. Society Ltd.,
Girgaum, Mumbai 400004, Maharashtra; Mobile : 9320129587
Tel : (022) 2385 6339 / 2386 9976, Fax : (022) 2386 9976
Email : niralimumbai@pragationline.com

☞ DISTRIBUTION BRANCHES

JALGAON

Nirali Prakashan : 34, V. V. Golani Market, Navi Peth, Jalgaon 425001,
Maharashtra, Tel : (0257) 222 0395, Mob : 94234 91860

KOLHAPUR

Nirali Prakashan : New Mahadvar Road, Kedar Plaza, 1st Floor Opp. IDBI Bank, Kolhapur 416 012,
Maharashtra. Mob : 9850046155; Email : niralikolhapur@pragationline.com

NAGPUR

Nirali Prakashan : Above Maratha Mandir, Shop No. 3, First Floor,
Rani Jhanshi Square, Sitabuldi, Nagpur 440012, Maharashtra
Tel : (0712) 254 7129; Email : niralinagpur@pragationline.com

DELHI

Nirali Prakashan : Room. No. 2 Ground Floor, 4575/15 Omkar Tower,
Agarwal Road, Darya Ganj, New Delhi - 110002.
Mob : 9555778814 / 9818561840 - Email : niralidelhi@pragationline.com

BENGALURU

Nirali Prakashan : Maitri Ground Floor, Jaya Apartments, No. 99, 6th Cross, 6th Main,
Malleswaram, Bangaluru 560 003, Karnataka. Mob : +91 9449043034
Email: niralibangalore@pragationline.com
Other Branches : Hyderabad, Chennai

Note: Every possible effort has been made to avoid errors or omissions in this book. In spite this, errors may have crept in. Any type of error or mistake so noted, and shall be brought to our notice, shall be taken care of in the next edition. It is notified that neither the publisher, nor the author or book seller shall be responsible for any damage or loss of action to any one of any kind, in any manner, therefrom. The reader must cross check all the facts and contents with original Government notification or publications.

Preface ...

We have great pleasure to present this book on **'Embedded System Design'** which is written as per the revised syllabus of Savitribai Phule Pune University for S.Y.B.Sc. (Computer Science) in the subject of Electronics. The authors have sincerely put their efforts to give complete information of the subject as prescribed in the syllabus in a simple manner.

Electronics and computer science go hand in hand. In fact, electronics is a backbone for understanding computer operations. As the technology is changing day by day, yesterday's technology becomes old today. Keeping this in mind, SPPU has introduced latest programming languages and technology in the syllabus.

This text book has been prepared as per the need of subject and syllabus specified by SPPU.

The First Chapter introduces the concept of Single Board Computer (SBC) and the fundamentals of Embedded System. The advantages/disadvantages of both are discussed in this chapter.

In Second Chapter the architecture of System on Chip (SoC) and the details of Raspberry Pi were thoroughly described. Raspberry Pi CPU, its architecture, internal blocks and their functions are also discussed.

Third Chapter is based on Python. The structure of Python code, variables, data types, functions, modules, packages used in Python were discussed in detail. Also few arithmetic programs in Python were explained in this chapter.

Fourth Chapter describe the interfacing of input output devices to Raspberry Pi. Their Python codes were also explained in this chapter.

After studying all this, student will get an idea of Raspberry Pi board and use of Python programming and also they will understand the concept of SBC and SoC.

To give something extra to think and to explore the topic, 'Think Over It' element is added in this book.

We are thankful to the publishers Shri Dineshbhai Furia and Shri Jignesh Furia and the staff of Nirali Prakashan specially Mr. Ilyas Shaikh, Mrs. Manasi Pingle, Ms. Chaitali Takle, Mr Ravindra Walodare for the great efforts they have taken to publish this book in time.

All valuable suggestions from the readers of this book are always welcome.

PUNE **AUTHORS**

JANUARY 2021

✍ ✍ ✍

Syllabus ...

1. **Introduction to Embedded Systems Using Single Board Computers (SBC) (08 Lectures)**

   Single boards computer block diagram, Types, Comparison of SBC models, Specifications, I/O devices (Storage, display, keyboard and mouse), Network access devices.

2. **Architecture of System on Chip (SOC) (08 Lectures)**

   Architecture of SoC, Basic version Broad Coprocessor, Pin Description of Raspberry Pi, Architectural features: CPU Overview, CPU Pipeline stages, CPU Cache Organization, Branch Prediction and Folding (Concept), GPU Overview.

3. **Programming Using Python (10 Lectures)**

   Overview of Rasberian OS (Operating System), Installation, different types of Operating Systems

   Basic Python Programming (Script programming): Variable and data types, Flow control structures, Conditional statements (If … Then … else),

   Functions: I/O function (GPIO, Digital), Time functions, Library functions.

   Basic Arithmetic Programs: Addition, Subtraction, Multiplication, Division.

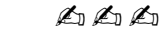
4. **Interfacing of Devices Using Python Programming (10 Lectures)**

   Basic interfacing: LED, Switch, LCD.

   Internal Advanced: Bluetooth, Wi-Fi, Ethernet.

   External advanced: Camera, Serial communication GSM, Ultrasonic Sensor, PIR, Finger Print reader.

Contents ...



Unit **1**...

Introduction to Embedded Systems Using Single Board Computers (SBC)

Charles Stark Draper

Charles Stark "Doc" Draper was an American scientist and engineer, known as the "father of inertial navigation". He was the founder and director of the Massachusetts Institute of Technology's Instrumentation Laboratory, later renamed the Charles Stark Draper Laboratory, which made the Apollo Moon landings possible through the Apollo Guidance Computer it designed for NASA. Draper was one of the pioneers of inertial navigation, a technology used in aircraft, space vehicles, and submarines that enables such vehicles to navigate by sensing changes in direction and speed, using gyroscopes, and accelerometers.

A pioneering figure in aerospace engineering, he contributed to the Apollo space program with his knowledge of guidance systems. One of the first recognizably modern embedded systems was the Apollo Guidance Computer, developed ca. 1965 by Charles Stark Draper at the MIT Instrumentation Laboratory. He was born on 2 October 1901,Windsor, Missouri, United States and died on 25 July 1987, Cambridge, Massachusetts, United States. He was awarded with National Medal of Science for Behavioral and Social Science, National Medal of Science for Engineering.

Introduction

The 8051 microcontroller, its architecture, instruction set, programming in assembly language, in 'C' language and interfacings of various input/output device were thoroughly studied in last semester. This semester, we have to study embedded system, its block diagram, its architecture, programming using Python and interfacing of devices using Python programming.

This chapter includes the details of embedded systems, its definition, block diagram, comparison of SBC models, Specifications, I/O devices (Storage, display, keyboard and mouse), Network access devices.

1.1 Embedded System

- An embedded system is a computer system which can perform many tasks such as to access, to process, to store, to control the data in various electronic systems.
- Embedded system is a combination of hardware and software.
- Software is usually known as firmware that is embedded into the hardware.
- Embedded systems are application specific, organized hardware which can be controlled by specific software.

- One of the important features of the embedded systems is that it gives the output within the specified time limits.
- Embedded systems are used in many appliances which are used in our daily life. It is used in microwave, calculators, TV, remote control, home security systems, traffic control system etc.
- The hardware of embedded system mainly consists of power source, microcontroller/microprocessor, timers, memory, I/O devises etc.
- The software consists of compliers, integrated development environment (IDE), assembler, simulators etc.
- Usually, embedded C, embedded C++, embedded JAVA, assembly language are used in embedded system programming.
- The block diagram of embedded systems is shown in Fig. 1.1.
- As shown in Fig. 1.1, embedded system consists of processor, program memory, data memory, processor, power supply, parallel ports, serial communication ports, I/O interfacing devices and application specific circuits.

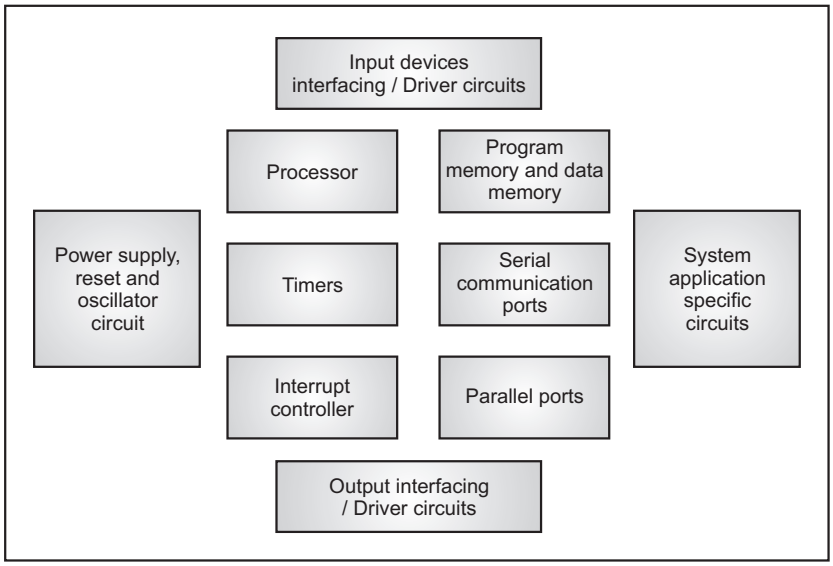


Fig. 1.1: Block diagram of Embedded system

- As mentioned earlier, embedded systems consists of hardware and software designed for a specific application.
- The illustration of basic structure of embedded system is shown in Fig. 1.2.

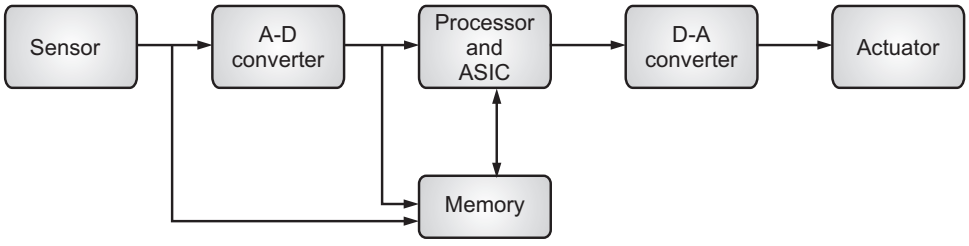


Fig. 1.2: Basic structure of Embedded system

- As shown in Fig. 1.2, embedded system is consists of sensor, ADC (Analog to Digital Converter), processor, DAC (Digital to analog Converter), actuators, memory etc.
- Sensor measures the physical quantity and converts it into an electrical signal. The output of a sensor is usually in analog form.
- ADC converts this analog output received by sensor into digital signal.
- Processor processes the data, calculates the output and stores it in memory.
- As this data has to be transmitted for further process or action, it has to be converted in to analog form again.
- DAC converts this digital output into analog form.
- Actuator compares the data with the expected output which is stored in memory and then further action is taken by the actuator.

Characteristics of Embedded Systems

- Embedded system usually performs a specialized operation.
- Embedded system must be of size to fit on a single chip.
- It must perform fast enough to process the data in real time.
- It has consumed less power to extent battery life.
- It has to monitor the data without any delay to give real time outputs.
- Microprocessor or microcontroller is used as a processor in embedded system.
- Memory has to be sufficient to store its software.
- It should contain peripherals to connect input/output devices.
- Software must be flexible and secure with more features.

Advantages of Embedded System:

- Low power consumption.
- Low cost.
- Small size.
- Enhanced real time performance.
- Easily customizable for a specific application .

Disadvantages of Embedded System

- High development cost.
- Time consuming design process.
- As its application specific, less market available.

1.2 Single Board Computers (SBC)

- A single board computer or SBC is a whole computer constructed on a single circuit board having memory, microprocessor, I/O devices and other features which are required for a functional computer.
- Personal computers have add-on slots in which various add-on cards can be inserted. However, SBC does not have any extra slots for I/O devices.

- Some SBC have plugs into backplane for system expansion.
- SBC usually use static RAM, low cost 8 bit microprocessor and few I/O/ devices.

Advantages of SBC:

- Easy to use.
- It has verified hardware.
- SBC has adaptability.
- Good performance at low price.
- Portable
- SBC has GPIO (General purpose input output) capability to interact with outside world.
- Low power consumption.
- Good support available for most of the SBC.
- SBC has small form factor and good computational quality.

Disadvantages of SBC:

- Usually delicate (electrically).
- If SBC is designed for an application or product which is in the high capacity category then it is cost effective otherwise it may cost more as compared with SOC (system on chip).
- Lot of customization on SBC can be difficult.
- SBC are not as powerful as the personal computers (PC). However, due to fewer components on single board, SBC usually consume less power and so do not dissipate much heat. Therefore, SBC in the smart phone can be used for entire day without charging and without cooling it.
- All electronic gadgets such as smart phone, tablets contain SBC. Single Board Computers are frequently used in embedded applications.
- An embedded computer cannot be expanded beyond its capability of I/O devices. For example, a vending machine can have an embedded single board computer to control all the functions of vending machine, but there is no provision to add more hardware to expand the system.
- In many cases, SBCs are plugged into a backplane which allows for input/output devices to be attached to the computer.
- SBC have all the capacity required for most of the automation tasks.
- SBC generally have less capacity than a multi-board computer.
- SBC are specialized equipment designed for a specific application. So, SBC are not manufactured in large quantities as compared to multi-board computers.
- SBCs are mainly used in industrial applications.

1.2.1 Single Board Computer Block Diagram

Block diagram of SBC is shown in Fig. 1.3 The main blocks of SBC are Processor, memory, General Purpose Input/Output (GPIO) pins, Ethernet port, External USB ports, microSD slot,

composite video/audio output, power supply, status LEDs etc. As per the application, the slots can vary.

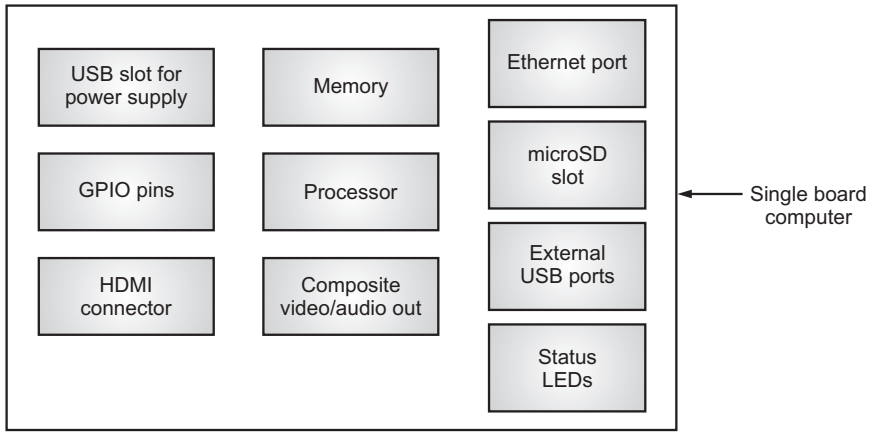


Fig. 1.3: Block diagram of SBC

- A Single Board Computer (SBC) is a complete computer built on a single circuit board as shown in Fig. 1.3.
- Basic blocks of SBC are shown in Fig. 1.3.
- The heart of the SBC is processor. Usually ARM processors are used in SBC.
- Static RAM (8 bit/16 bit) is mostly used in SBC.
- SBC contains GPIO header to interface input and output devices such as sensors, displays, keyboards etc.
- External USB ports are provided to interface other devices.
- Ethernet port is also provided to connect other devices to SBC.
- HDMI port provides digital video and audio output.
- MicroSD slots are given on SBC to increase the available memory.
- Usually SBC operates on 5V Dc power supply which can be connected through micro USB slot.
- SBC can have various extra ports and slots as per the need or application.

1.2.2 Types of SBC

- There are two basic broad categories of SBCs; those which contain passive backplane and other are standalone SBCs.
- Backplane SBC has the advantage of a standard interface for expansion cards such as ISA, PCI etc.
- Standalone SBCs also has expansion cards which are less standardized. Backplane SBCs works with various architectures including Intel architecture.
- Single Board Computers are used in variety of application. SBCs are designed for a specific application for example, Raspberry Pi which was developed as an educational tool to help, encourage and strengthened the programming skills of students.

- The BeagleBoard and BeagleBone were also developed to help, educate and promote the benefits and usage of open source hardware/software in embedded computing.
- Specifications of SBCs vary as per the application. Some of the SBCs come with I/O interfaces required for audio applications, networking applications or wireless applications. Some of the SBC contains expansion slots for PCI, ISA, VGA etc.
- SBC uses various types of processors such as ARM, Intel, other power architectures. Each processor family has its own attributes that appeal to a specific application. ARM processor is based on RISC (Reduced Instruction Set Computer) architecture has digital signal processing capabilities. ARM processor is very popular for its low power consumption. Intel processor is mainly used in defense and aerospace applications. Power architecture is based on RISC which supports legacy upgrades.
- Most of the SBCs use Linux operating system.
- Popular examples of SBCs are Raspberry Pi, BeagleBoard, BeagleBone, PandaBoard, Cubieboard, Marsboard, Hackberry, Udoo, APC 8750, Intel Thin etc.
- Important Factors considered while selecting SBC are power, form factor, backward pin compatibility, processor type, memory, operating system, I/O devices, overall performance.

1.2.3 Comparison of SBC Models

- Single Board Computers are used in most of the electronic gadgets such as smart phones, tablets.
- Some notable SBCs are available in the market for the development of hardware and software such as Raspberry Pi, Beagles (BeagleBoard, BeagleBoardxM, BeagleBone, BeagleBone Black), PandaBoard, MK802, MK808, Cubieboard, MarsBoard, Hackberry, Udoo, DragonBoard. Intel Joule, nVidia Jetson etc. Recently, Intel has introduced Atom processor based on MinnowBoard.
- However, out of all these types, Raspberry Pi and The Beagles/BeagleBone Black are most cost effective and small sized SBCs.
- Below section describes the specifications and details of I/O devices of these SBCs. Features of few SBCs are given below:

1. BCM5871X

- The BCM5871X is a series of quad-core 64-bit ARM processor specifically designed for networking applications and NFV applications, control panel processing for Ethernet switches.
- Network attached storage (NAS) etc. The BMC5871X has advanced computing, networking and virtualization functions on a single SoC (System-on chip) with good performance and power efficiency.

Features:

- Quad-core ARM Cortex.
- Hardware virtualization support.

- Advanced power management mode.
- Support for open source projects and development models.

2. TI AM3358 based on ARM Cortex-A8

- The TI AM3358 processors offer a cost effective solution for industry applications. These processors combine interface peripherals and display capabilities.
- The AM3358 is an ARM Cortex microprocessor with image graphics, peripherals and industrial interface options. It supports high level operating systems, Linux, Android etc. It also supports real-time responses, specialized data handling operations.

Features:

- 32 bit RISC processor with 64 kB RAM.
- External memory interfaces.
- General purpose memory controller.
- Error locator model.
- Power, reset and clock management.

3. Intel Atom T5500

- Intel Atom T5500 and T5700 are quad core Atom processor based on Intel Goldmont architecture.
- It contains new 18EU Intel Gen9 graphics and media GPU with quick sync technology for 4K video encoding and decoding. It includes various peripherals related to IoT.

Features:

- Contains sensor Hub processor.
- Has power management.
- USB 3.0 and 2.0 ports.
- 12C, UART, PWM and GPIOs.
- Up to 6x MIPI CSI cameras.
- Integrate WiFi and Bluetooth through PCI Express interface.

4. NVIDIA

- NVIDIA is the ARM Cortex A15 based quad core CPU having very good performance and widely used in mobile applications.
- It contains second generation battery saver CPU core. It is power efficient. NVIDIA is the first multi-core mobile system-on-chip processor (SoC) introduced in 2011.

Features:

- Has full featured Web browsing.
- Console class gaming.
- Significantly high performance.
- More efficient user interface.

5. Snapdragon 410

- Snapdragon 410 is 64 bit ARM Cortex A53 SoC mostly used in mobile phones.
- It is quad core processor and has Adreno 306 graphics card, memory controller, Bluetooth, UMT/LTE radio etc.

The comparison of above SBCs is given in below table.

Table 2.1 : Comparison of SBCs

Processor	Broadcom BCM2837 based on ARM Cortex-A53	TI AM3358 based on ARM Cortex-A8	Intel Atom T5700	nVidia 4-plus-1 based on ARM Cortex-A15	Qualcomm Snapdragon 410C based on ARM Cortex-A53
Architecture	64 bit RISC	32 bit RISC	64 bit x86	32 bit RISC	64 bit RISC
Cores	Quad cores	Single core	Quad core	Quad core	Quad core
Clock Speed	1.2 GHz	1 GHz	1.7 GHz burst to 2.4 GHz	2.3 GHz	1.2 GHz
GPU	Broadcom VideoCore IV	Power VR SGX530	Intel HD	nVidia Kepler GPU	Aualcomm Adreno 306
Memory Type	LPDDR2	DDR3	LPDDR4	DDR3L	LPDDR3
Memory	1 GH	512 MB	4 GB	2 GB	1 GB
Onboard Storage	No	4 GB eMMC Flash	16 GB eMMC Flash	16 GB eMMC Flash	8 GB eMMC Flash
Supported Operating Systems	Android, Various Linux Distributions, Windows 10 IoT Core	Android, Various Linux Distributions	Ubuntu Desktop/ Core, Ostro/Yocto Linux, Windows 10 IoT Core	Linux 4 Tegra (Ubuntu), Android	Android, Various Linux Distributions Windows 10 IoT Core
Wifi facility	Yes	No	Yes	No	Yes
Bluetooth facility	Yes	No	Yes	No	Yes
GPIO	40 possible with remapping	69 possible with remapping	8 Dedicated/ 48 when remapping interface pins	8 Dedicated	12
GPIO Voltage	3.3 V	3.3 V	1.8 V	1.8 V	1.8 V
PWM	2	8	4	4	0
USB	4 × Type A	1 × Type A-mini Client, 1 × Type A Host	1 × Type C 3.1, 1 × Type A 3.0	1 × Type A-Micro, 1 × Type A 3.0	1 × Type A-micro, 2 × Type A 2.0

... (Contd.)

PCIe	No	No	1 Port - 5GB/s	Half Mini-PCIE slot	No
Camera Interface	Yes 1 × MIPI-CSI	Yes 1 × GPMC	Yes 2 × MIPI-CSI	Yes 2 × MIPI-CSI, x4 and x1	Yes 1 × MIPI-CSI
HDMI Output	1080p60	1080p24	1080p60	2160p30	1080p30
Display Interface	HDMI	micro HDMI	HDMI, MIPI-DSI (4K)	HDMI, LVDS	HDMI, MIPI-DSI
4K Video Encode	No	No	4k × 2kp60 (H.264), 4kx2kp30 (MVC), 1080p60 (VP8), 1080p30 (VP9)	4kx2kp24 (H.264)	1080p30 (H.264)
4K Video Decode	No	No	4kx2kp60 (H.264, MVC, VP8, VP9)	4kx2kp30 (H.264)	1080p30 (H.264), 720p (H.265)
Power Consumption	250 - 750 mA @ 5V	210 - 640 mA @ 5V	130 - 600 mA, @ 12V	250 - 4800 mA @ 12V	50 - 400 mA @ 12V

1.2.4 Comparison of Raspberry Pi and BeagleBone

- Raspberry Pi is a Single Board computer developed in United Kingdom by Raspberry Pi Foundation. It is specially developed for educational purpose. It is high performance computer with low cost. Raspberry Pi is used in development of the projects such as retro gaming arcade, cryptocurrency wallet, home theatre PC etc.

Features:

- Quad core 64 bit ARM Cortex A53 processor.
- 4 USB ports.
- HDMI video outputs.
- VideoCore IV multimedia.

BeagleBone:

- BeagleBone is a low cost, community supported development platform for all learners.
- It is tiny open-software which can be used for any type of small projects. BeagleBone Black model has mechanical and header compatibility.

Features:

- 3D graphics.
- NEON Floating point processor.
- 5V Dc External Via Expansion Header.

- Optional onboard serial header.
- Has Wifi, Bluetooth.

- BeagleBone Black has faster processor and RAM, internal flash storage, DC power jack, an excellent OS support. Raspberry Pi is specially developed for educational purpose.
- It is a teaching/learning resource. BeagleBone is useful for developers. Raspberry Pi is suitable for the applications which involve media centre or GUI. BeagleBone is suitable for embedded systems or robotics.
- In next chapter Raspberry Pi, its pin configuration, I/O devices and all other details will be discussed.

1.3 Input / Output Devices

- Single board computers (SBC) contains less number of circuits on the board by reducing connectors and bus driver circuits that has to be used in PCs. SBC consists of input output devices which are inbuilt or can be connected through connectors.
- Some of the SBC has slots in which I/O devices can be connected.
- Embedded SBCs contains units providing all required I/O with no provision for plug in cards.
- Usually SBCs are plugged into a backplane to provide support to I/O cards.

Some of the standard input and output devices which are commonly used in embedded systems/SBC are discussed in this section.

1.3.1 Memory

- Basic types of memories are primary and secondary and then RAM (Random Access Memory) and ROM (Read only Memory) as shown in below Fig. 1.4.

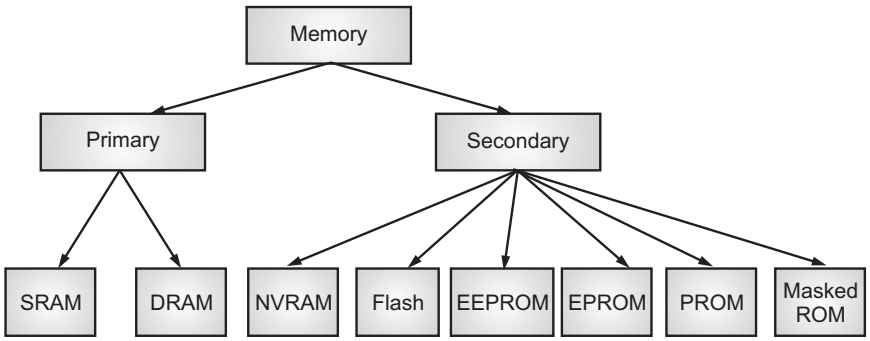


Fig. 1.4: Types of Memories

- As shown in Fig. 1.4, types of primary memory are SRAM (Static Random Access Memory) and DRAM (Dynamic Random Access Memory).
- Types of secondary memories are Masked ROM, Non-volatile RAM, Flash memory, Programmable ROM (PROM), Erasable Programmable ROM (EPROM) and Electrically Erasable Programmable ROM (EEPROM).

- Usually embedded system consists of tiny memory chips from 128 MB or 32MB. SBCs usually contain flash memory from 2 GB to 32 GB. Some of them have memory expansion slots.

1.3.2 Keyboard and Mouse

- This is one of the input devices. A keyboard is an array of switches which can include some internal logic circuit to interface the embedded systems.
- There are various types of keyboards used in embedded systems.
- The keyboards can be interfaced to SBC through USB port or on backplane. Some of them are discussed below.

(i) Membrane Keyboard:

- In membrane keyboard, the keys are cushioned with rubber or silicone shell. These keys are very sensitive to the pressure applied on it.
- Membrane keyboard consists of three layers such as a top membrane layer, an inner middle layer and a bottom membrane layer.
- A conductive which is in between these two layers gets activated when a pressure is applied on it. This type of keyboard can work better with high speed of typing.
- Membrane keyboards are available at reasonable cost. However, if keys are not hit properly with sufficient pressure then keys are not registered. This is the major issue of membrane keyboard.
- Another problem is due to rubber dome membrane. There is a rubber dome membrane which supports all the keys. This membrane is designed to resist depression. However, due to this membrane, the keys may not resume their original position after user withdraws his finger from the key.

(ii) Mechanical Keyboards:

- Mechanical keyboards are very popular due to its durability and responsiveness (accuracy). They are expensive compared with membrane keyboards.
- Key switches are used in mechanical keyboards by which keys hit a keystroke and physically trigger a switch that sends a signal to register the key press.
- Working mechanism of mechanical keyboards and membrane keyboard is similar.
- Mechanical keyboard also resists keystroke that is the keys return to their initial position once a user removes their fingers. In these types of keyboards, it uses springs instead of rubber membrane. Springs result in smoother and more accurate keystrokes. This feature makes mechanical keyboard more superior and popular.
- It also supports high typing speed with long durability. However, mechanical keyboards produce more noise during use as compared with membrane keyboards.

(iii) Wireless Keyboards:

- The main benefit of wireless keyboards is that it reduces wiring and can be operated from a distance.
- Two main technologies are used in wireless keyboards, Bluetooth and radio-frequency (RF).

- Bluetooth is the most popular technology used to interface the keyboard to SBC. It has limited range of communication. Radio- frequency keyboards utilize a broadcast technique similar to Wi-fi and can be interfaced to SBC. Through USB dongle these keyboards can be attached with SBC.
- Wireless keyboards need recharging periodically, modern keyboards usually can work for months.

(iv) Rollup USB Keyboards:

- Nowadays, rollup USB wired keyboards are available in market. These keyboards can blend easily and further can be rolled.
- They are small in size and light in weight. Due to its compact structure, they are becoming popular in travelling professionals.
- Rollup keyboards are similar to the basic keyboard, only their manufacturing is little bit different than the basic keyboard.
- Rollup keyboards are generally waterproof. They are designed to resist spills and immersion in water. This keyboard produces very low noise which can be barely detected.

(v) Projection Keyboards:

- Projection keyboards are also known as Optical Virtual keyboard. They are latest innovation in the keyboard technology.
- They usually use lasers to display a visible virtual keyboard onto a surface.
- When a user selects a key on the virtual keyboard, then optical sensor or camera senses the finger movement which is translated it into actions. Then Computer software identifies the key and further the character pressed by the user is recognized.

(vi) TouchScreen:

- Touchscreen is an input/output device. It can be used as keyboard and also as a display.
- The touchscreen registers the position of a touch on its surface and accordingly displays the information on the screen.
- Two types of touchscreens resistive and capacitive are usually used in SBC.
- A resistive touchscreen uses a two dimensional voltmeter to sense the position on the screen. It consists of two conductive sheets separated by spacer balls. The top conductive sheet is flexible so that it can touch the bottom sheet. A voltage is applied across the sheet whose resistance causes a voltage gradient to appear across the sheet. An analog to digital converter is used to measure the voltage and resulting position. The resistive touchscreen is usually used in industrial applications and can be suitable for roughest surroundings. Resistive touchscreen are also operable with gloves and touchscreen pencils.
- Capacitive touchscreen displays are mostly used at places where a very high light transmission is needed. The coating of SiO_2 makes the surface of capacitive touchscreen very scratch resistance. So they are usually used for applications at public places.

- These are the few types of keyboards which are commonly used in embedded systems or in SBCs. The keyboards which are used in PCs are more expensive and use a microprocessor to identify the key press. The major advantage of using microprocessor is that it can provide debouncing of the keys.

1.3.3 Display

Display is an output device. Usually, monitors are used as a display device in Personal Computers.

(i) LCD :

- In SBC or in embedded systems usually LCD is used as a display which can be interfaced through a slot. Size of LCD display is selected as per the application.
- Liquid Crystal Display (LCD) consists of several layers, polarized panel filters and electrodes. The rod shaped tiny molecules are sandwiched between a flat piece of glass and an opaque substrate. When electric charge is applied to it, these rod shaped molecules align into two different physical positions. When electric charge is applied they align to block the light entering thorough them and if charge is not applied then they become transparent. When light passes through it, the desired image appears on it. LCD is thin, flat and consumes very less energy as compared with LED displays of CRTs (Cathode Ray Tubes).
- Types of LCD are based on displayed data such as Segment LCD, Graphical LCD, Colour LCD etc. Segment LCD displays numbers, letters and fixed symbol. They are usually used in industrial panels. Graphical LCD has pixels in rows and columns. By energizing a particular set of pixels, any character can be displayed. Color LCD has two types such as Passive Matrix and Active Matrix. Passive color LCD is simple to implement as compared with Active color LCD. Active color LCD is sharp and has better viewing angle.
- Main specifications of LCD are resolution, viewable size, response time, refresh rate, matrix type, viewing angle, contrast angle, aspect ratio etc.
- LCDs are used in SBC or in embedded systems as they consumes very less power, they are small in size, low weight, flat, portable, no flicker, less screen glare which reduces eyestrain. However, LCDs cannot display multiple resolution images. The contrast ratio for LCD images is lesser than CRT and plasma display.
- LCD has longer response time because of which ghost images can appear or images can be mixed.
- LCD also has narrow viewing angle which can reduce image quality. But for most of the industrial applications LCDs are used as a display.

(ii) LED :

- Light Emitting Diode (LED) display consists of array of LEDs to emit red, green and blue light which forms color images. LED display mainly used for outdoor big screens, traffic lights, in TVs etc.

(iii) Plasma :

- Plasma display is also flat display. Plasma display emits light by itself. Each pixel of plasma display is illuminated by a tiny bit of plasma or charged gas. The mercury vapor is inserted into vacuum glass tube. When voltage is applied, the gas generates the effect of plasma and emits ultraviolet light which excites the phosphor to generate visible light. Each pixel is generated by three different colors (Red, Green, Blue) which creates color image.

1.4 Network Access Devices

- Single Board Computers and embedded systems use various network access devices for the wired and wireless communication. Three types of embedded networking devices which are widely used are discussed in this section.

(i) Ethernet:

- Ethernet is the standard networking device used to connect computers on a network using wired connection.
- It is simple interface and also used to connect SBC to other devices.
- A single router and a few Ethernet cables can create a Local Area Network (LAN) which connects devices to communicate to each other.
- Ethernet can operate in a small area such as a room, office or cabin.
- Ethernet is a network protocol that controls data transmission of LAN. When any device connected in the network wanted to send the data to another device, it senses the carrier on the connecting wires. If it is free, then it sends the data packet on the network. Other device check the packet, if it is the expected receiver then it receives data.
- Ethernet connects any computer or any electronic device to its network through Ethernet adapter or network card.
- A network interface card either integrated into the motherboard of computer or on SBC or separately installed in the device. For Ethernet networking, a router, hub, switch or gateway is required. A hub is a device that acts as a connecting point between devices on a network. Cable is unshielded twisted pair (UTP) cables used in Ethernet LAN. These cables are similar to the cables which are used for landline telephone. Ethernet can be managed on Windows, Linex or macOS.
- Most Ethernet devices are backward compatible and can be used with lower speed Ethernet cables and devices.
- Ethernet is still the standard for wired networking. However, today it has been replaced by wireless networks such as Wi-fi, Bluetooth etc.

(ii) CAN Bus :

- The Control Area Network (CAN) protocol is specially developed for automotive industry.
- CAN is a serial network which can be used to establish local connections between microcontroller in a motor vehicle.
- CAN bus protocol is a two-wire, half duplex system which is suitable for high speed transfer of short messages.

- CAN was basically developed to reduce cable wiring and separate electronic control units inside the vehicle can communicate to each other.
- CAN is low cost and durable network. The major advantage of CAN is that electronic control unit can have single CAN interface rather than having multiple analog /digital inputs to every device. Thus, CAN reduce overall cost and weight of a system in automobiles.
- In CAN, all the devices have CAN controller chip and so all devices can see all transmitted messages. Each device can decide whether the message is relevant or should be filtered. Devices can be added in CAN without any modification in the network.
- Every message in CAN has a priority. So if two devices try to send message simultaneously, the one with highest priority can only transmit the message.
- CAN also include Cyclic Redundancy Check (CRC) code for error checking of all received messages. Frames or messages with errors are disregarded by all the nodes or devices and error frame can be transmitted to indicate the error in the network.
- CAN is mostly used in automobile industry. However, because of various advantages, today CAN is used in various applications such as streetcars, light railways, long distance trains, navigation systems, medical equipments, automatic doors etc.

(iii) I2C :

- I2C is a two wire, half duplex serial communication protocol with multi-master, multi-slave architecture.
- I2C is used to interface low speed devices like microcontrollers, EEPROMs, A/D, D/A, I/O peripheral or in similar embedded devices. It was invented by Philips.
- I2C bus is popular because it is simple and can have more than one master. Each slave device has a unique address. Communication between master and slave is serial. So data can be transferred bit by bit.
- I2C is used for single boards and also used to connect various components which are linked via cable. I2C requires only two bus lines for communication.
- I2C can work with any baud rate. Simple master-slave relationship exists between all components in I2C. Each device has a unique address. I2C also include collision detection and provide arbitration.
- There are various other network devices used in SBC or in embedded systems. Most popular and widely used network access devices and protocols are discussed in this section.

Think Over It

- List the differences between microcontroller and SBC.
- Can Arduino called as SBC? Explain.
- Can SoC called as SBC?

Points to Remember

- An embedded system is a computer system which can perform many tasks such as to access, to process, to store, to control the data in various electronic systems.
- Embedded system is a combination of hardware and software.
- One of the important features of the embedded systems is that it gives the output within the specified time limits.
- The hardware of embedded system mainly consists of power source, microcontroller/microprocessor, timers, memory, I/O devises etc.
- A single board computer or SBC is a whole computer constructed on a single circuit board having memory, microprocessor, I/O devices and other features which are required for a functional computer.
- SBC usually use static RAM, low cost 8 bit microprocessor and few I/O/ devices.
- There are two basic broad categories of SBCs; those which contain passive backplane and other are standalone SBCs.
- Backplane SBC has the advantage of a standard interface for expansion cards such as ISA, PCI etc.
- Standalone SBCs also has expansion cards which are less standardized. Backplane SBCs works with various architectures including Intel architecture.
- Popular examples of SBCs are Raspberry Pi, BeagleBoard, BeagleBone, PandaBoard, Cubieboard, Marsboard, Hackberry, Udoo, APC 8750, Intel Thin etc.
- Raspberry Pi is a Single Board computer developed in United Kingdom by Raspberry Pi Foundation. It is specially developed for educational purpose. It is high performance computer with low cost.
- BeagleBone is a low cost, community supported development platform for all learners.
- Liquid Crystal Display (LCD) consists of several layers, polarized panel filters and electrodes.
- Main specifications of LCD are resolution, viewable size, response time, refresh rate, matrix type, viewing angle, contrast angle, aspect ratio etc.
- Light Emitting Diode (LED) display consists of array of LEDs to emit red, green and blue light which forms color images.
- Plasma display emits light by itself. Each pixel of plasma display is illuminated by a tiny bit of plasma or charged gas.
- Ethernet is the standard networking device used to connect computers on a network using wired connection.
- Most Ethernet devices are backward compatible and can be used with lower speed Ethernet cables and devices.
- The Control Area Network (CAN) protocol is specially developed for automotive industry.
- CAN bus protocol is a two-wire, half duplex system which is suitable for high speed transfer of short messages.
- I2C is a two wire, half duplex serial communication protocol with multi-master, multi-slave architecture.

Exercises

[A] True or False :

1. A single board computer or SBC is a whole computer constructed on a single circuit board having memory, microprocessor, I/O devices and other features which are required for a functional computer.
2. The Control Area Network (CAN) protocol is specially developed for chemical industry.
3. Mechanical keyboard has more durability compared with other type of keyboards.
4. BeagleBone is a low cost, community supported development platform for all learners.
5. Usually ARM processors are used in SBC.

[B] Multiple Choice Questions :

1. Which of the following is NOT a disadvantage of embedded systems?
   (a) High development cost (b) Time consuming design process
   (c) Less market available (d) Small size
2. Ethernet port is provided to connect other devices to SBC. True or False?
   (a) True (b) False
3. Which of the following is a type of SBC?
   (a) Toshiba (b) BeagleBone
   (c) Intel (d) Sony
4. LED consumes more energy than LCD. True or False?
   (a) True (b) False
5. CAN protocol is specifically designed for _____ industry.
   (a) Pharmaceutical (b) Home automation
   (c) Automobiles (d) Farming

[C] Fill in the Blanks:

1. Embedded system is a combination of _____ and _____.
2. The disadvantage of embedded system is _____ development cost.
3. HDMI port provides digital ____ and _____ output.
4. ____ slots are given on SBC to increase the available memory.
5. Most of the SBCs use _____ operating system.
6. BeagleBone Black is a type of ____ .

[D] Short Answer Questions:

1. What is an embedded system?
2. What is SBC?
3. State any two characteristics of embedded system.
4. State any two features of SBC.
5. State any two advantages and disadvantages of embedded system.
6. Define SBC.

7. Why SBC are popular and preferred in some of the applications?
8. State any types of SBC.
9. Compare any two SBCs.
10. What are the different types of memories?
11. State any two types of keyboards.
12. Write types of network access devices of embedded system or SBC.

[E] Long Answer Questions:

1. Draw and explain the block diagram of single board computer.
2. List characteristics of embedded system.
3. Draw and explain the block diagram of embedded system.
4. What are the advantages and disadvantages of SBC?
5. Explain any two types of SBC in detail.
6. Compare any three types of SBC.
7. Explain types of memory in detail.
8. Explain membrane keyboard.
9. Explain mechanical keyboard.
10. What are different types of keyboards?
11. Write a short note on LCD display.
12. Explain Ethernet in detail.
13. What id I2C protocol?
14. Explain CAN protocol.

Answers

[A] True or False :

(1) True (2) False (3) True
(4) True (5) True

[B] Multiple Choice Questions :

1. (d)	2. (a)	3. (b)	4. (b)	5. (c)

[C] Fill in the Blanks :

(1) Hardware and Software (2) High (3) Video and Audio
(4) MicroSD (5) Linux (6) SBC

Unit 2...

Architecture of System on Chip (SOC)

Eben Christopher Upton

Eben Christopher Upton is born on 5th April 1978 and is a British technical director and ASIC architect for Broadcom. He is also a founder and former trustee of the Raspberry Pi Foundation, and now CEO of Raspberry Pi (Trading) Ltd.

He has completed a Bachelor of Arts degree in Physics and Engineering in 1999 at the University of Cambridge . Then he went on to do the Cambridge Diploma in Computer Science graduating in 2001.

After his diploma, Upton was a research student in the Computer Laboratory, University of Cambridge. After finishing his Ph.D. he took a Master of Business Administration (MBA) at the Cambridge Judge Business School while working in industry.

Upton was Director of Studies in Computer Science at St John's College, Cambridge, with responsibility for undergraduate admissions. During his academic career, he co-authored papers on mobile services, Human–computer interaction (HCI), Bluetooth and data dependency graphs. He has been a visiting Researcher at Intel Corporation, Founder and Chief Technology Officer at Idea works 3D and a software engineer at IBM.

Introduction

- A System-on-chip (SoC) is an integrated circuits (IC) which has most of the components of a computer or any electronic system. These components include CPU (Central Processing Unit), memory, I/O devices, graphical processing unit (GPU) etc.
- SoC is a complete electronic system which may contain analog, digital or radio frequency functions. SoC is specially designed to meet the standards of required electronic circuits of various computer components onto a single substrate or chip. Thus, instead of assembling several chips and components onto a circuit board, the SoC fabricates all required circuits into one unit.
- Unlike the motherboard based PC architecture which has various components on a circuit board, SoC integrates all these components on a single chip.
- As SoC is an IC which contains both hardware /software, it consumes less power and has better performance. It has small size and it is more reliable than multi chip systems.
- Best examples of SoC are smart phones, Internet of Things, embedded systems, tablets, mobile computing, edge computing etc.
- There are some issues with SoC such as higher prototyping and architecture costs, it is more complex to debug. It is not cost effective and requires more time to manufacture.

Advantages of SoC:

The basic purpose of using SoC is to have all components on a single chip with minimum components.

The advantages of SoC are:

- Size of SoC is small and it includes many features and functions.
- SoC consumes less power.
- SoC is flexible in terms of chip size, power, form factor (form factor refers to the size, shape, and physical specifications of hardware components) .
- As SoC is build on a single chip for a specific application, it is cost effective if produced in large quantity.
- As SoCs are highly integrated, hard cores can be used to implement highly computational functions in hardware.

Disadvantages of SoC:

- The designing process of SoC usually takes six to twelve months. So it is time consuming to have SoC for a specific application.
- If any component on SoC is not functioning properly, then it cannot be replaced. In that case, entire SoC has to be replaced.
- Visibility of SoC is limited.

In previous chapter, Single Board Computer (SBC) was thoroughly studied. Here the **differences between SoC and SBC** are listed as:

- SBC is a whole computer constructed on a single printed circuit board (PCB) which contains memory, processor, I/O and other slots. SoC is an integrated circuit (IC) or silicon chip which has all the components fabricated on silicon chip.
- Many peripherals are connected to SBC through slots given on it. SoC have all the components integrated on it. So SBCs are more adaptable as compared with SoC. One single mistake while designing the SoC can be very expensive as it is a chip.
- SBC and SoC are entirely different to each other. SoC is a significant component of SBC.
- SoCs are widely used in industry as they have small form factor, less power consumption and small size. SBCs are used to advance the end product.
- SBC has inbuilt hardware and software which includes SoCs, memory, connectivity interfaces, USB, CAN, HDMI, video/audio inputs etc.

2.1 Architecture of SoC

SoC is a small chip which contains all required components and circuits of a particular system.

Usually, SoC contains memory, application processor, voltage regulators, power management circuits, timing sources such as oscillators, external interface for USB, Universal Asynchronous Receiver Transmitter (UART), RAM/ROM, ADC/DAC, operating system etc. The basic block diagram of SoC is shown in Fig. 2.1.

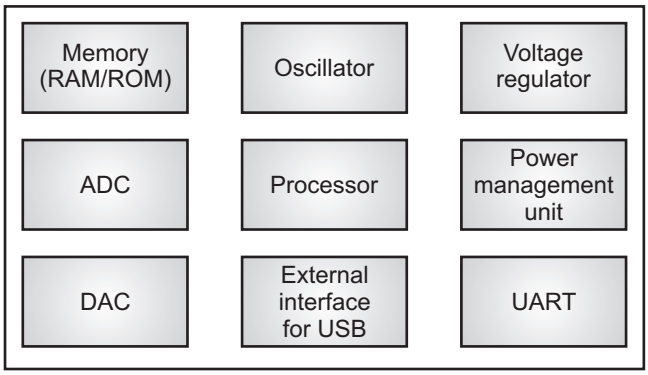


Fig. 2.1 : Block diagram of SoC

- Basic architecture of SoC is shown in Fig. 2.1. Processor is the heart of SoC. Usually, SoC has multiple core processors. It can be a microcontroller, a microprocessor or digital signal processor or an application specific instruction set processor.
- SoC contains memory for storage. It may have RAM, ROM, EEPROM or flash memory.
- SoC include external interfaces such as USB, Ethernet, HDMI or can have Wi-fi and Bluetooth which help to meet the industry standard communication protocol.
- SoC also has GPU (Graphical Processing Unit) to visualize the interface.
- Voltage regulators, oscillators, clocks, ADC/DAC are also part of SoC.
- Universal Asynchronous Receiver Transmitter (UART) is included which is used to transmit or receive serial data.
- SoC has internal interface bus or network to connect all the individual blocks on SoC.

As mentioned earlier, SoC is a silicon chip having all the components fabricated on it which makes its size very small and also reduces wiring. Nowadays, SoCs are becoming popular because of its application, small size and low power consumption.

2.2 Basic Version Board Coprocessor

Raspberry Pi has Basic version board processor, the details of which are discussed below.

Raspberry Pi:

Raspberry Pi is a small single board computer (SBC). Raspberry Pi foundation is registered as an UK based Educational Charity. The aim of the foundation is to provide low cost, high performance computer capable of creating, modeling specific applications.

- Raspberry Pi is credit size computer which can be plugged into a monitor. It acts as minicomputer by connecting keyboard, mouse and display.
- Raspberry Pi 3 is based on Broadcom System on Chip with an ARM processor.
- Raspberry Pi is widely used for real time image/video processing, robotics, IoT based applications.
- Raspberry Pi foundation officially provides Debian based Raspbian OS and NOOBS OS. Raspbian OS is freely available OS. Raspbian have GUI (Graphical User Interface) which includes tools like browsing, python programming, games etc.
- Raspberry Pi provides access to GPIO for developing various applications such as LED, motors, sensors etc.
- Raspberry Pi provides on-chip I2C, UART, I2S modules.

2.2.1 Architecture and Pin Configuration of Raspberry Pi

Raspberry Pi has ARM processor and 512MB RAM. The architecture of Raspberry Pi and its pin description are discussed in this section. Below Fig. 2.2 shows the architecture of Raspberry Pi.

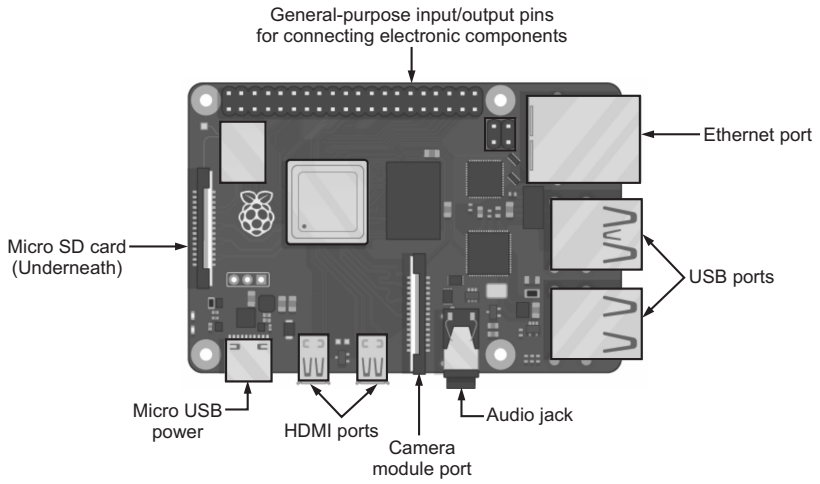


Fig. 2.2 : Raspberry Pi architecture

Source: https://projects.raspberrypi.org/en/projects/raspberry-pi-getting -started/2

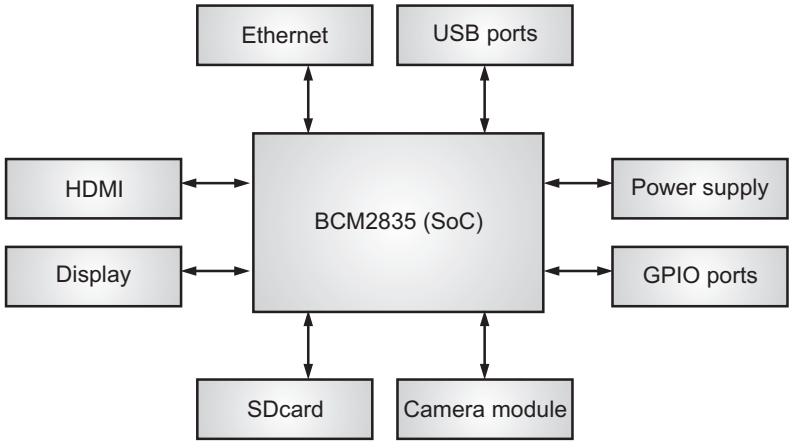


Fig. 2.3 : Main blocks of Raspberry Pi

Raspberry Pi mainly consists of following blocks:

- **Processor:** Raspberry Pi uses Broadcom BCM2835 System on Chip which is ARM processor and Videocore Graphics Processing Unit (GPU). It is the heart of the Raspberry Pi which controls the operations of all the connected devices and handles all the required computations.
- **HDMI:** High Definition Multimedia Interface is used for transmitting video or digital audio data to computer monitor or to digital TV. This HDMI port helps Raspberry Pi to connect its signals to any digital device such as monitor or digital TV or display through HDMI cable.
- **GPIO ports:** General Purpose Input Output ports are available on Raspberry Pi which allows the user to interface various I/O devices.
- **Audio output:** Audio connector is available for connecting audio output devices such as headphones, speakers.

- **USB ports:** This is a common port available for various peripherals such as mouse, keyboard or any other I/O device. With the help of USB port, the system can be expanded by connecting more peripherals.
- **SD card:** SD card slot is available on Raspberry Pi. An SD card with an operating system (OS) installed is required for booting the device.
- **Ethernet:** Ethernet connector allows access to wired network. It is available only on the Model B of Raspberry Pi.
- **Power supply:** Micro USB power connector is available onto which a 5V Power supply can be connected.
- **Camera module:** Camera Serial Interface (CSI) connects Broadcom processor to Pi camera.
- **Display:** Display Serial Interface (DSI) is used for connecting LCD to Raspberry Pi using 15 pin ribbon cable. DSI provides high resolution display interface which is specifically used for sending video data.

2.2.2 Pin Description of Raspberry Pi

A row of GPIO (General Purpose Input Output) pins is available on Raspberry Pi. A 40 pin header is provided on Raspberry Pi as shown in below Fig. 2.4.

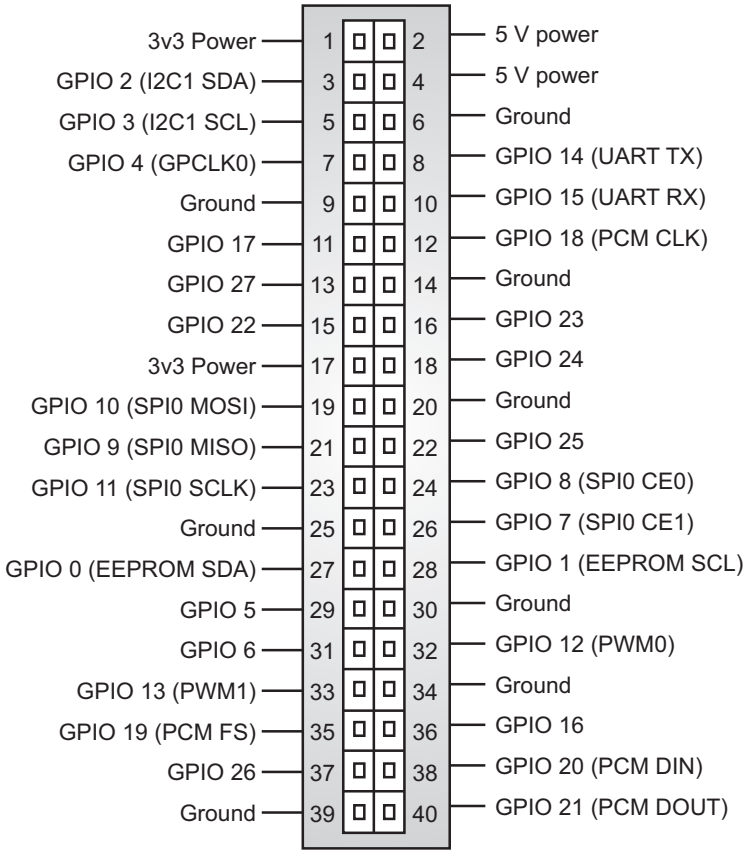


Fig. 2.4 : 40 pin header on Raspberry Pi

- Any of GPIO pins can be used as an input or output pin if they are designated in the software. The output pin can be set to high (3.3 V) or low (0 V). GPIO pin designated as an input pin can be read as high (3.3 V) or low (0 V).
- Two 5 V (pin no. 2 & 4) and two 3.3 V (pin no. 1 & 17) pins are available on the Raspberry Pi board. Also a number of ground pins (pin no. 6, 9, 14, 20, 25, 30, 34, 39) are available. Remaining pins are all general purpose 3.3 V pins, that is outputs are set to 3.3 V and inputs are 3.3 V tolerant.
- **GPIO 2 and GPIO 3 pins (pin no. 3 & 5) :** SDA (I2C1 & 2) are pins of the I2C on the Pi. SDA includes a fixed 1.8 kΩ pull-up to 3.3 V. That is, these pins are not suitable for use as a general purpose I/O where no pull-up resistor is desired.
- **GPIO 14 & GPIO 15:** These pins (pin no. 8 & 10) are used as transmit and receive pins for UART on Pi.
- **GPIO 0 & GPIO 1:** These pins (pin no. 27 & 28) are generally kept reserved for I2C communication with an EEPROM.
- **GPIO 7 & GPIO 8:** These pins (pin no. 24 & 26) are used as Special Peripheral Interface (SPI) to communicate with other peripherals.
- **GPIO 12 & GPIO 13:** Hardware Pulse Width Modulation (PWM) are available on these pins (pin no. 32 & 33).
- **GPIO 19:** Pin number 35 is used by PCM to provide a frame-sync signal to an external audio device such as DAC.
- **GPIO 20:** Pin number 38 is used by PCM as data input from audio device such as an I2C microphone.
- **GPIO 21:** Pin number 40 is used by PCM to provide a data output signal to an external audio device such as DAC.

The brief description of all the pins of Raspberry Pi is tabulated in below table:

Table 2.1 : Pin Description of Raspberry Pi

Pin group	Pin name	Description
Power Source	+5 V, +3.3 V, GND and Vin	+5 V - power output +3.3 V - power output GND – GROUND pin
Communication Interface	UART Interface (RXD, TXD) [(GPIO 15,GPIO 14)]	UART (Universal Asynchronous Receiver Transmitter) used for interfacing sensors and other devices.

… ***(Contd.)***

Pin group	Pin name	Description
SPI Interface (MOSI, MISO, CLK,CE) × 2 [SPI 0-(GPIO 10, GPIO 9, GPIO 11, GPIO 8)] [SPI 1--(GPIO 20, GPIO 19, GPIO 21, GPIO 7)]	SPI (Serial Peripheral Interface) used for communicating with other boards or peripherals.	Other devices can be connected serially at these pins.
TWI Interface (SDA, SCL) × 2 [(GPIO 2, GPIO 3)] [(ID_SD,ID_SC)]	TWI (Two Wire Interface) Interface can be used to connect peripherals.	SDA, SCL etc. can be connected at these pins.
INPUT OUTPUT PINS	26 I/O	Although these some pins have multiple functions they can be considered as I/O pins.
PWM	Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19	These 4 channels can provide PWM (Pulse Width Modulation) outputs. *Software PWM available on all pins.
EXTERNAL INTERRUPTS	All I/Os	In the board all I/O pins can be used as Interrupts.

There are different versions of Raspberry Pi available today as listed below:

1. Raspberry Pi 1 Model A
2. Raspberry Pi 1 Model A+
3. Raspberry Pi 1 Model B
4. Raspberry Pi 1 Model B+
5. Raspberry Pi 2 Model B
6. Raspberry Pi 3 Model B
7. Raspberry Pi Zero

The features of Raspberry Pi models which are widely used today are given in below Table 2.2.

Table 2.2 : Features of Raspberry Pi Models

Features	Raspberry Pi Model B+	Raspberry Pi 2 Model B	Raspberry Pi 3 Model B	Raspberry Pi zero
SoC	BCM2835	BCM2836	BCM2837	BCM2835
CPU	ARM11	Quad Cortex A7	Quad Cortex A53	ARM11
Operating Frequency	700 MHz	900 MHz	1.2 GHz	1 GHz
RAM	512 MB SDRAM	1 GB SDRAM	1 GB SDRAM	512 MB SDRAM
GPU	250 MHz Videocore IV	250 MHz Videocore IV	400 MHz Videocore IV	250 MHz Videocore IV
Storage	Micro-SD	Micro-SD	Micro-SD	Micro-SD
Ethernet	Yes	Yes	Yes	No
Wireless	WiFi and Bluetooth	No	No	No

2.3 Architectural Features of Raspberry Pi

As mentioned earlier, Raspberry Pi has various modules with advanced features. The architectural features of Raspberry Pi are discussed in this section.

The Main Features of Raspberry Pi:

The main features of Raspberry Pi are listed as:

- First generation of Raspberry Pi consists of the quad core Broadcom BCM2835 SoC. It includes ARM1176JZF-S processor, VideoCore 4 graphics processing unit (GPU) and 1 GB RAM.
- BCM43438 wireless LAN and Bluetooth on board, 100 Base Ethernet.
- 40 pin extended GPIO.
- 4 USB 2 ports.
- 4 pole stereo output and composite video port.
- Full size HDMI.
- CSI camera port for connecting a Raspberry Pi camera.
- DSI display port for connecting a Raspberry Pi touchscreen display.
- Micro SD port for loading operating system and storing data.
- Micro USB power source.

The peripheral of BCM2835 and CPU used in Raspberry Pi is thoroughly discussed in below section.

2.4 Peripherals used in BCM2835

- Raspberry Pi is a series of SBCs developed for people who are interested in basic computer science. It gained popularity due to its usage in robotics, its small size and its functionality.
- The first generation of Raspberry Pi used BroadCom2835 SoC. All models of Raspberry Pi consist of Broadcom SoC.
- Broadcom SoC consists of ARM compatible CPU with on-chip GPU, VideoCore IV.
- The speed of CPU ranges from 700 MHz to 1.2 GHz.
- The on board memory ranges from 256 MB to 1 GB RAM. SD cards are used to store the operating systems.
- Most of the Raspberry Pi boards have one to four USB slots, HDMI composite video output and an audio jack.
- Lower level output is provided by GPIO pins which support I2C protocol. The recent Raspberry B model has 8P8C Ethernet port and the Pi 3 and Pi Zero have Wi-fi 802.11 and Bluetooth on board.
- The first generation of Raspberry Pi has 256 MB RAM, next generation doubled the RAM whereas third generation doubled it further.
- Raspberry Pi 3 Model released in February 2016 consists of on board Wi-fi and Bluetooth.

2.5 Raspberry Pi Processor

- The first generation of Raspberry Pi used BroadCom2835 SoC. The Raspberry Pi 2 uses BCM2836. The third generation of Raspberry Pi used BCM2837 SoC.
- Over the years, the performance of Raspberry Pi goes on increasing. First generation Raspberry Pi provides a performance similar to 300 MHz Pentium II, third generation of Raspberry Pi increases its performance 10 times than first generation raspberry Pi.
- The details of BroadCom2835 are given below. The block diagram of BCM2835 is shown in below Fig. 2.5.

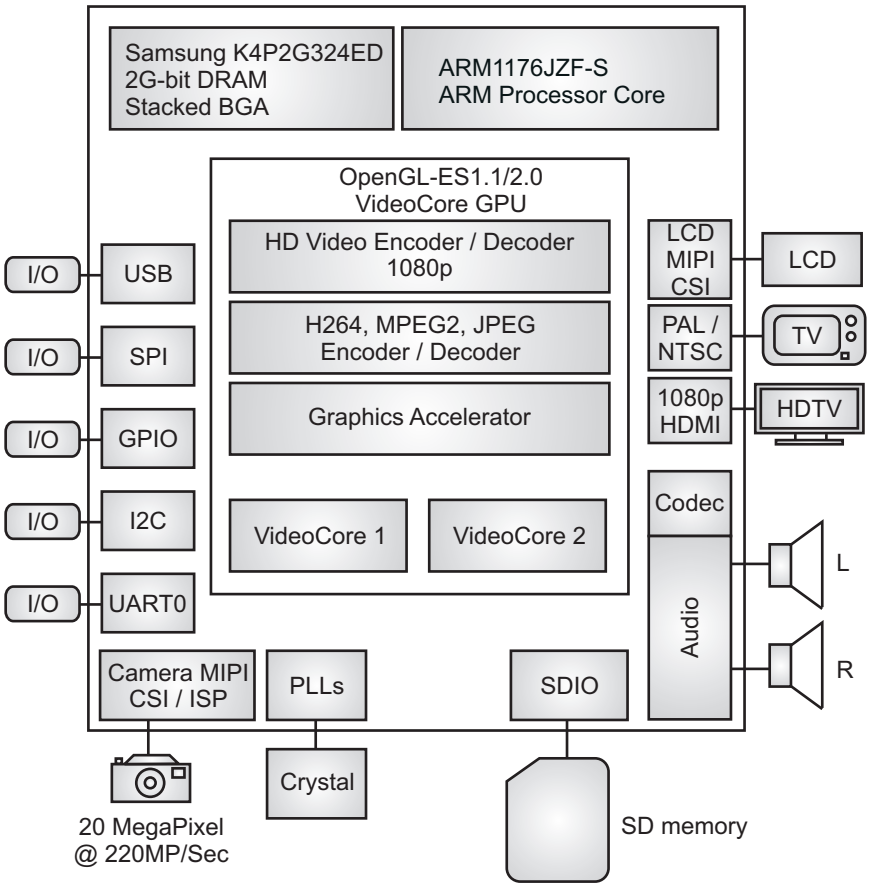


Fig. 2.5: Block diagram of BCM2835

- The BroadCom2835 is a System on Chip (SoC). It contains multimedia capabilities which are used in mobile phones and portable devices.
- It provides full HD video and stereo audio.
- It contains VideoCore IV multimedia coprocessor, ARM1176JZ-F application processor and GPU.

- It also contains advanced Image Sensor Pipeline (ISP) for connecting 20 MP camera.
- It contains Timers, Interrupt controller, GPIO, USB, PCM, I2C, DMA controller, I2C/SPI slave, SPIO, PWM, UART0, UART1.
- The BCM 2835 has 3 Auxiliary peripherals; one mini UART and two SPI masters. They share same area in the peripheral and have common interrupts. They are controlled by auxiliary enable register.
- Two auxiliary registers controls interrupt status register, auxiliary enable register.
- Mini UART has to be enabled before use. Mini UART has following features as:
    - 7 or 8-bit operation.
    - One start bit and one stop bit.
    - No parities.
    - Break generation.
    - 8 symbols deep FIFOs for receive and transmit data.
    - Auto flow control with programmable FIFO level.
    - 16550 registers.
    - Baud rate derived from system clock.
- However, as it is mini UART, it does not have following:
    - Break detection.
    - Framing error detection.
    - Parity bit.
    - Receive time-out interrupt.
    - DSR (Data Set Ready), DTR (Data Terminal Ready), RI (ring Indicator) signals.
- Two SPIs (Special Peripheral Interface) have to be enabled before use. They have following features:
    - Single bit length between 1 and 32-bit.
    - Multi beat infinite bit length.
    - Three independent chip select per master.
    - 32 bit wide transmit and receive FIFOs.
    - Wide clocking range.
    - Programmable data out hold time.
    - Shift in-out MS or LS bit first.
    - Clock inversion (idle high or idle low).

However, there are some issues with SPI. There is no SPI standard in any form. Some SPI interfaces are widely used with memory devices, so their rules are followed. The universal SPI has been developed to work with the non-standard SPI devices.

- The BroadCom Serial controller (BSC) is a master, fast mode (400 kb/s) BSC controller. BSC is compatible to I2C bus.
- Majority of the hardware pipelines and peripherals of BCM2835 are bus master which manages their own data.
- DMA (Direct Memory Access) controllers are directly connected to the peripherals. The BCM2835 DMA controller provides 16 DMA channels.
- There are 54 general purpose I/O (GPIO) lines split into two banks. All GPIO have at least two alternative functions. The GPIO peripherals have three dedicated interrupt lines. These lines are triggered by the setting of bits in the event detect status register.
- The ARM has two types of interrupt sources; interrupt coming from the GPU peripherals, interrupt coming from local ARM control peripherals and special events interrupts. For each interrupt source (ARM or GPU) there is an interrupt enable bit (read/write) and an interrupt pending bit (read only).
- The PCM audio interface provides high quality serial audio streams. It supports many classic PCM formats including I2S.
- Pulse Width Modulator (PWM) controller have features as: (a) two independent output bit streams, clocked at a fixed frequency (b) bit streams configured individually to output of PWM or to serialized version of 32-bit word (c) PWM output have variable input and output resolutions.
- Serial Interface Peripherals (SPI) have features as: (a) implement three wire aerial protocol SPI or SSP (Synchronous Serial Protocol) (b) provide supports for polled interrupt or DMA operation (c) implements LoSSI Master (Low Speed Serial Interface).
- The BCM2835 has two UARTs (Universal Asynchronous Receiver/Transmitter). The UART performs serial to parallel conversion of data characters received from external peripheral devices or modem. The UART provides separate 16×8 transmit and 16×12 receive FIFO memory. The UART contains standard asynchronous communication bits (start, stop, parity). However, DSR, DTR and RI are not supported.

BCM2835 uses ARM16JZF-S processor as CPU. The architecture, explanation of each block and its features are discussed in next section.

2.6 CPU Architecture

Raspberry Pi uses ARM116JZF-S CPU. The CPU architecture of ARM116JZF-S is shown in below Fig. 2.6.

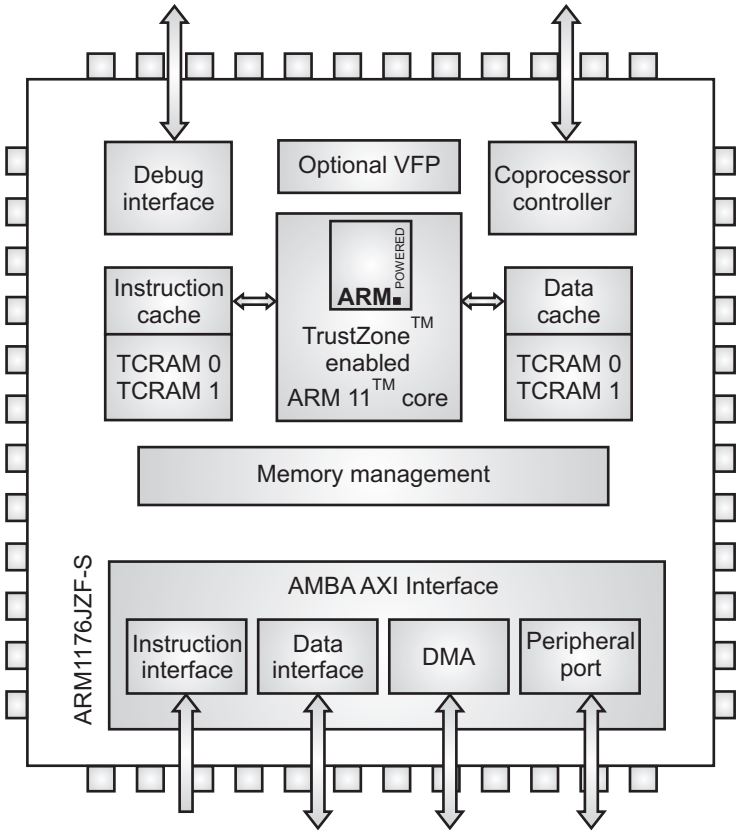


Fig. 2.6: CPU architecture of ARM116JZF-S

- The ARM1176JZ-S processor is an integer core which implements ARM11. It supports Java bytecodes.
- The features of ARM1176JZ-S are:
  - Provision for Intelligent Energy Management (IEM).
  - High speed Advanced Microprocessor Bus Architecture (AMBA), Advanced Extensible Interface (AXI) which supports prioritized multiprocessor implementations.
  - An Integer core with integral Embedded ICE-RT logic.
  - An eight stage pipeline.
  - Branch prediction with return stack.
  - Low interrupt latency configuration.
  - Internal coprocessor CP14 and CP15.
  - External coprocessor interface.

- Instruction and data Memory Management Units (MMUs), managed using MicroTLB structures backed by Main TLB.
- Virtually indexed and physically addressed caches.
- 64-bit interface to both caches.
- Tightly-Coupled Memory (TCM) which can be used as local RAM with DMA.
- External coprocessor support, trace support.
- JTAG-based debug.
- ARM1176JZF-S processor includes Vector Floating Point (VFP) coprocessor whereas ARM1176JZ-S does not include VFP. Other features of both the processors are same.
- The ARM1176JZ-S processor has three instruction sets as : 32-bit ARM instruction set used in ARM state with media instructions, 16-bit Thumb instruction set used in Thumb state and 8-bit JAVA bytecodes used in Jazelle state.

Components of the ARM1176JZ-S Processor:

The block diagram of ARM1176JZ-S processor is shown in Fig. 2.7.

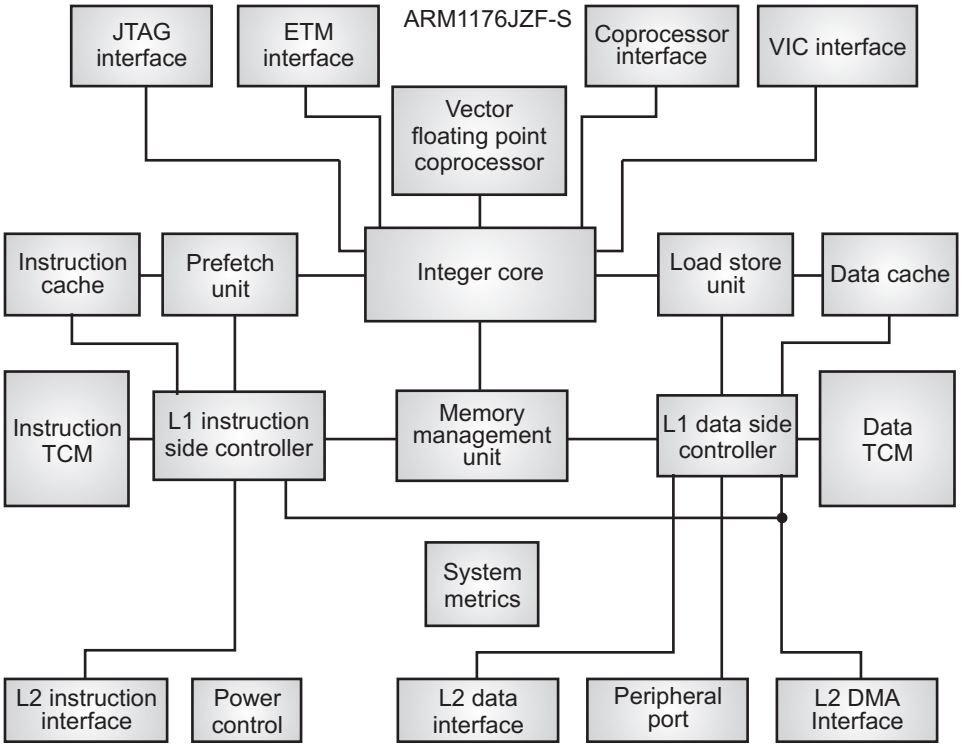


Fig. 2.7: Block diagram of ARM1176JZ-S

The **main components** of the **ARM1176JZ-S** processor are:

1. Integer core
2. Load store unit (LSU)
3. Prefetch unit
4. Memory system

5. AMBA AXI interface
6. Coprocessor interface
7. Debug
8. Instruction cycle summary and interlocks
9. Vector Floating Point (VFP) coprocessor
10. System control
11. Interrupt handling

1. Integer Core:

- The ARM1176JZ-S has AARM11 Integer Core. The processor contains Embedded CE-RT logic and a JTAG debug interface which enables the hardware debuggers to communicate with the processor. The Integer core consists of following sections:
  (i) Instruction Set category
  (ii) Conditional execution
  (iii) Registers
  (iv) Modes and exceptions
  (v) Thumb instruction set
  (vi) DSP instructions
  (vii) Media extension
  (viii) Datapath
  (ix) Branch prediction
  (x) Return stack

All sections of integer core are discussed below.

(i) Instruction set categories:

The main instruction set categories are:

1. Branch instructions
2. Data Processing instructions
3. Status register transfer instructions
4. Load and store instructions
5. Coprocessor instructions
6. Exception-generating instructions.

Only load, store, and swap instructions can access data from memory.

(ii) Conditional execution: The processor conditionally executes almost all the ARM instructions. The flags such as condition code flag, negative or sign, zero, carry and overflow flags will get updated after each arithmetic or logical instructions.

(iii) Registers: The ARM1176JZ-S core contains 33 general purpose 32-bit registers and 7 are dedicated 32-bit registers. At any time only 16 general purpose registers are visible. Remaining registers are banked registers used to speed up execution processing.

(iv) Modes and exceptions: The ARM1176JZ-S processor provides a set of operating modes to support complex operating systems, user applications and real-time demands. There are 8 operating modes out of which 6 are exception processing modes:

- User
- Supervisor
- Fast interrupt
- Normal interrupt
- Abort
- System
- Undefined
- Secure monitor.

(v) Thumb instruction set: The Thumb instruction set contains a subset of the 32-bit ARM instruction encoded into 16-bit wide opcodes. This reduces the amount of memory required for instruction storage.

(vi) DSP instructions: The DSP extensions to the ARM instruction set provide:

- 16-bit data operations.
- Saturating arithmetic.
- MAC operations.

The processor executes multiply instructions using a single cycle 32×16 implementation. The processor can perform 32×32, 32×16 and 16×16 multiply instructions (MAC).

(vii) Media extension: The ARMv6 instruction set provides media instruction to complement the DSP instructions. There are 4 media instruction groups:

- Multiplication instructions for handling 16-bit and 32-bit data, including dual-multiplication instructions that operate on both 16-bit halves of their source registers. This group includes an instruction that improves the performance and size of code for multi-word unsigned multiplications.
- *Single Instruction Multiple Data* (SIMD) instructions to perform operations on pairs of 16-bit values held in a single register, or on sets of four 8-bit values held in a single register. The main operations supplied are addition and subtraction, selection, pack, and saturation.
- Instructions to extract bytes and halfwords from registers, and zero-extend or sign-extend them. These include a parallel extraction of two bytes followed by extension of each byte to a halfword.
- Unsigned *Sum-of-Absolute-Differences* (SAD) instructions. These are used in MPEG motion estimation.

(viii) Datapath: The Datapath consists of 3 pipelines: (a) ALU, shift and Sat pipeline, (b) MAC pipeline, (c) Load/store pipeline

(a) ALU, Shift and Sat pipeline:

The ALU, Shift and Sat pipeline executes most of the ALU operations, and includes a 32-bit barrel shifter. It consists of three pipeline stages:

ALU: The ALU stage performs all arithmetic and logic operations and generates the condition codes for instructions that set these flags. The ALU stage consists of a logic unit, an arithmetic unit and a flag generator. The pipeline logic evaluates the flag settings in parallel with the main adder in the ALU. The flag generator is enabled only on flag-setting operations. The ALU stage separates the carry chains of the main adder for 8-bit and 16-bit SIMD instructions.

Shift: The shift stage contains the full barrel shifter. Shift stage performs all shifts, including those required by the LSU. The shift stage implements saturating left shift that doubles the value of an operand and saturates it.

Sat: The Sat stage implements the saturation logic required by the various classes of DSP instructions.

(b) MAC pipeline: The MAC pipeline executes all of the enhanced multiply and multiply-accumulate instructions.

The MAC unit consists of a 32×16 multiplier and an accumulate unit that is configured to calculate the sum of two 16×16 multiplies. The accumulate unit has its own dedicated single register read port for the accumulate operand. To minimize power consumption, the processor only clocks each of the MAC and ALU stages when required.

(c) Load/store pipeline: Load/store pipeline is a part of the Load/Store Unit (LSU). LSU encompasses the entire L1 data side memory system and the integer load/store pipeline. This includes:

- the L1 data cache.
- the data side TLB.
- the integer store buffer.
- the NEON store buffer.
- the integer load data alignment and formatting.
- the integer store data alignment and formatting.

The pipeline accepts one load or store per cycle that can be present in either pipeline 0 or pipeline 1. This gives the processor flexibility when scheduling load and store instructions.

(ix) Branch prediction: The Integer Core uses both static and dynamic branch prediction. All branches are predicted where the target address is an immediate address or fixed-offset PC-relative address.

The first level of branch prediction is dynamic through a 128-entry *Branch Target Address Cache* (BTAC). If the PC of a branch matches an entry in the BTAC, the processor uses the branch history and the target address to fetch the new instruction stream.

The processor might remove dynamically predicted branches from the instruction stream and might execute such branches in zero cycles.

If the address mappings are changed then the BTAC must be flushed. A BTAC flush instruction is provided in the CP15 coprocessor.

The processor uses static branch prediction to manage branches not matched in the BTAC. The static branch predictor makes a prediction based on the direction of the branches.

(x) Return stack: The processor includes a three-entry return stack to accelerate returns from procedure calls.

For each procedure call, the processor pushes the return address onto a hardware stack. When the processor recognizes a procedure return, the processor pops the address held in the return stack that the prefetch unit uses as the predicted return address.

2. Load Store Unit (LSU):

- The Load/Store Unit (LSU) manages all load and store operations. The load/store pipeline decouples loads and stores from the MAC and ALU pipelines. When the processor issues LDM and STM instructions to the LSU pipeline, other instructions run concurrently subject to the requirements of supporting precise exceptions.

3. Prefetch Unit:

- The main purpose of Prefetch Unit (PFU) is to perform speculative fetch of instructions ahead of the DPU by predicting the outcome of branch instructions and to format instruction data in a way that aids the DPU in efficient execution.
- The PFU fetches instructions from the memory system under the control of the DPU and the internal coprocessors CP14 and CP15.
- In ARM state, the memory system can supply up to two instructions per cycle.
- In Thumb state, the memory system can supply up to four instructions per cycle.
- The PFU buffers up to three instruction data fetches in its FIFO. There is an additional FIFO between the PFU and the DPU that can normally buffer up to eight instructions. This reduces or eliminates stall cycles after a branch instruction. This increases the performance of the processor.

Program flow prediction occurs in the PFU by:

- Predicting the outcome of conditional branches using the branch predictor and for direct branches, by calculating their destination address using the offset encoded in the instruction.
- Predicting the destination of procedure returns using the return stack.
- The DPU resolves the program flow predictions that the PFU makes.
- The PFU fetches the instruction stream as dictated by:
  - The program counter.
  - The branch predictor.
  - Procedure returns signaled by the return stack.
  - Exceptions including aborts and interrupts signaled by the DPU.
  - Correction of mispredicted branches as indicated by the DPU.
- The PFU starts instruction fetches at a rate that is determined dynamically using a prediction scheme which aims to ensure that the pipeline is kept fed with instructions without over-fetching instructions that are not used.

- Fetching of unused instructions consumes extra power and can impact performance.
- The prefetch unit fetches instructions from the instruction cache, Instruction TCM, or from external memory and predicts the outcome of branches in the instruction stream.

4. Memory System:

The memory system provides the core with following:

- Separate instruction and data caches.
- Separate instruction and data tightly-coupled memories.
- 64-bit datapaths throughout the memory system.
- Virtually indexed, physically tagged caches.
- Memory access controls and virtual memory management.
- Support for four sizes of memory page.
- Two-channel DMA into TCMs.
- I-fetch, D-read/write interface, compatible with multi-layer AMBA AXI.
- 32-bit dedicated peripheral interface.
- Export of memory attributes for second-level memory system.

Sections in Memory System:

The memory system consists of following sections:

(i) Instruction and data caches.
(ii) Cache power management.
(iii) Instruction and data TCM.
(iv) TCM DMA engine.
(v) DMA features.
(vi) Memory management unit.

These sections are described in details as follows:

(i) Instruction and data caches:

The core provides separate instruction and data caches. The cache has the following features:

- Independent configuration of the instruction and data cache during synthesis to sizes between 4 kB and 64 kB.
- 4-way set-associative instruction and data caches. Each can be locked independently.
- Pseudo-random or round-robin replacement.
- Eight word cache line length.
- The MicroTLB entry determines whether cache lines are write-back or write-through.
- Ability to disable each cache independently, using the system control coprocessor.
- Data cache misses that are non-blocking. The processor supports up to three outstanding data cache misses.
- Streaming of sequential data from LDM and LDRD operations, and sequential instruction fetches.
- Critical word first filling of the cache on a cache-miss.

- All the cache RAM blocks and the associated tag and valid RAM blocks can be implemented using standard ASIC RAM compilers. This ensures optimum area and performance of the design.
- Each cache line is marked with a Secure or Non-secure tag that defines if the line contains Secure or Non-secure data.

(ii) Cache power management:

To reduce power consumption, the core uses sequential cache operations to reduce the number of full cache reads. If a cache read is sequential to the previous cache read and the read is within the same cache line, only the data RAM set that was previously read is accessed. The core does not access tag RAM during sequential cache operations.

To reduce unnecessary power consumption additionally, the core only reads the addressed words within a cache line at any time.

(iii) Instruction and data TCM:

Because some applications might not respond well to caching, configurable memory blocks are provided for Instruction and Data *Tightly Coupled Memories* (TCMs). These ensure high-speed access to code or data.

An Instruction TCM typically holds an interrupt or exception code that the processor must access at high speed, without any potential delay resulting from a cache miss.

A Data TCM typically holds a block of data for intensive processing, such as audio or video processing.

Each TCM can be configured to be Secure or Non-secure.

(iv) TCM DMA engine:

The size of the *Instruction TCM* (ITCM) and the size of the *Data TCM* (DTCM) can be separately configured to be 0 kB, 4 kB, 8 kB, 16 kB, 32 kB or 64 kB. For each side (ITCM and DTCM):

- If the TCM size is configured to be 4 kB, we can get one TCM of 4 kB, on this side.
- If the TCM size is configured to be larger than 4 kB, we get two TCMs on this side, each of half the configured size. So, for example, if an ITCM size of 16 kB is configured, we get two ITCMs, each of size 8 kB.

Below Table lists all possible TCM configurations.

Table 2.3 : TCM configurations

Configured TCM size	Number of TCMs	Size of each TCM
0 kB	0	0
4 kB	1	4 kB
8 kB	2	4 kB
16 kB	2	8 kB
32 kB	2	16 kB
64 kB	2	32 kB

The TCM can be anywhere in the memory map. The **INITRAM** pin enables booting from the ITCM.

To support use of the TCMs by data-intensive applications, the core provides two DMA channels to transfer data to or from the Instruction or Data TCM blocks.

DMA can proceed in parallel with CPU accesses to the TCM blocks. Arbitration is on a cycle-by-cycle basis.

The DMA channels connect with the *System-on-Chip* (SoC) backplane through a dedicated 64-bit AMBA AXI port.

The DMA controller is programmed using the CP15 system-control coprocessor. DMA accesses can only be to or from the TCM, and an external memory. There is no coherency support with the caches.

Only one of the two DMA channels can be active at any time.

(v) DMA features:

The DMA controller has the following features:

- Runs in background of CPU operations.
- Enables CPU priority access to TCM during DMA.
- Programmed with virtual addresses.
- Controls DMA to either the instruction or data TCM.
- Allocated by a privileged process (OS).
- Software can check and monitor DMA progress.
- Interrupts on DMA event.
- Ability to configure each channel to transfer data between Secure TCM and Secure external memory.

(vi) Memory Management Unit:

The *Memory Management Unit* (MMU) has a unified *Translation Lookaside Buffer* (TLB) for both instructions and data.

The MMU includes a 4 kB page mapping size to enable a smaller RAM and ROM footprint for embedded systems and operating systems such as Windows CE that have many small mapped objects.

The ARM1176JZ-S processor implements the *Fast Context Switch Extension* (FCSE) and high vectors extension that are required to run Microsoft Windows CE.

The MMU is responsible for protection checking, address translation, and memory attributes, and some of these can be passed to an external level two memory system.

The memory translations are cached in MicroTLBs for each of the instruction and data caches, with a single Main TLB backing the MicroTLBs.

The MMU has the following features:

- Matches Virtual Address, ASID and NSTID.

- Each TLB entry is marked with the NSTID.
- Checks domain access permissions.
- Checks memory attributes.
- Translates virtual-to-physical address.
- Supports four memory page sizes.
- Maps accesses to cache, TCM, peripheral port, or external memory.
- Hardware handles TLB misses.
- Software control of TLB.

Paging:

Four page sizes are supported by 16 MB super sections, 1 MB sections, 64 kB large pages and 4 kB small pages.

Domains:

Sixteen access domains are supported.

TLB:

A two-level TLB structure is implemented. Eight entries in the main TLB are lockable. Hardware TLB loading is supported and is backwards compatible with previous versions of the ARM architecture.

ASIDs:

TLB entries can be global or can be associated with particular processes or applications using *Application Space IDentifiers* (ASIDs).

ASIDs enable TLB entries to remain resident during context switches to avoid subsequent reload of TLB entries and also enable task-aware debugging.

NSTID:

TrustZone extensions enable the system to mark each entry in the TLB as Secure or Non-secure with the *Non-Secure Table IDentifier* (NSTID).

System Control Coprocessor:

Cache, TCM and DMA operations are controlled through a dedicated coprocessor CP15, integrated within the core.

This coprocessor provides a standard mechanism for configuring the level one memory system, and also provides functions such as memory barrier instructions.

5. AMBA AXI interface:

- The Advanced Microcontroller Bus Architecture (AMBA) protocols are an open standard, on-chip interconnect specification for the connection and management of functional blocks in a SoC.
- AMBA facilitates right-first-time development of multi-processor designs with large number of controllers and peripherals.
- AMBA bus interface provides high bandwidth connections between the processor, second level caches, on-chip RAM, peripherals, and interfaces to external memory.

There are separate bus interfaces for:

- Instruction fetch, 64-bit data.
- Data read/write, 64-bit data.
- Peripheral access, 32-bit data.
- DMA, 64-bit data.

- All interfaces are AMBA AXI compatible. This enables them to be merged in smaller systems. Additional signals are provided on each port to support second-level cache.

The ports support the following bus transactions:

(a) Instruction fetch: Servicing instruction cache misses and non-cacheable instruction fetches.

(b) Data read/write: Servicing data cache misses, hardware handled TLB misses, cache eviction and non-cacheable data reads and writes.

(c) DMA: Servicing the DMA engine for writing and reading the TCMs. This behaves as a single bidirectional port.

These ports enable several simultaneous outstanding transactions, providing:

- High performance from second-level memory systems that support parallelism.
- High use of pipelined and multi-page memories such as SDRAM.

The following sections describe the AMBA AXI interface in more detail:

(i) Bus clock speeds.
(ii) Unaligned accesses.
(iii) Mixed-endian support.
(iv) Write buffer.
(v) Peripheral port.

(i) Bus clock speeds:

The bus interface ports operate synchronously to the CPU clock if IEM is not implemented.

(ii) Unaligned accesses:

The core supports unaligned data access. Words and halfwords can align to any byte boundary. This enables access to compacted data structures with no software overhead. This is useful for multi-processor applications and reducing memory space requirements.

The *Bus Interface Unit* (BIU) automatically generates multiple bus cycles for unaligned accesses.

(iii) Mixed-endian support:

The core provides the option of switching between little-endian and byte invariant big endian data access modes. This means, the core can share data with big-endian systems, and improves the way the core manages certain types of data.

(iv) Write buffer:

All memory writes take place through the write buffer. The write buffer decouples the CPU pipeline from the system bus for external memory writes. Memory reads are checked for dependency against the write buffer contents.

(v) Peripheral port:

The peripheral port is a 32-bit AMBA AXI interface that provides direct access to local, Non-shared devices separately.

The peripheral port does not use the main bus system. The memory regions that these non-shared devices use are marked as Device and Non-Shared.

Accesses to these memory regions are routed to the peripheral port instead of to the data read-write ports.

6. Coprocessor Interface:

- The ARM1176JZ-S processor connects to external coprocessors through the coprocessor interface.
- This interface supports following ARM coprocessor instructions:
  - LDC
  - LDCL
  - STC
  - STCL
  - MRC
  - MRRC
  - MCR
  - MCRR
  - CDP
- The memory system returns data for all loads to coprocessors in the order of the accesses in the program. The processor suppresses HUM operation of the cache for coprocessor instructions.
- The external coprocessor interface relies on the coprocessor executing all its instructions in order.
- Externally-connected coprocessors follow the early stages of the core pipeline to permit the exchange of instructions and data between the two pipelines. The coprocessor runs one pipeline stage behind the core pipeline.
- To prevent the coprocessor interface introducing critical paths, wait states can be inserted in external coprocessor operations. These wait states enable critical signals to be retimed.
- Coprocessor interface describes the interface for on-chip coprocessors such as floating-point or other application-specific hardware acceleration units.

7. Debug:

- The ARM1176JZ-S core implements the ARMv6.1 Debug architecture that includes extensions of the ARMv6 Debug architecture to support TrustZone.

- It introduces three levels of debug as:
  - debug everywhere,
  - debug in Non-secure privileged user and Secure user,
  - debug in Non-secure user only.
- The debug coprocessor, CP14, implements a full range of debug features that *Debug* and *Debug Test Access Port* describe.
- The core provides extensive support for real-time debug and performance profiling.

The following sections describe debug in more detail:

(i) System performance monitoring
(ii) ETM interface
(iii) ETM trace buffer
(iv) Software access to trace buffer
(v) Real-time debug facilities
(vi) Debug and trace environment.

(i) System performance monitoring:

This is a group of counters that one can configure to monitor the operation of the processor and memory system.

(ii) ETM interface:

One can connect an external *Embedded Trace Macrocell* (ETM) unit to the processor for real-time code tracing of the core in an embedded system.

The ETM interface collects various processor signals and drives these signals from the core.

The interface is unidirectional and runs at the full speed of the core.

The ETM interface connects directly to the external ETM unit without any additional glue logic. The ETM interface can be disabled for power saving.

(iii) ETM trace buffer:

The functionality of the ETM can be extended by adding an on-chip trace buffer. The trace buffer is an on-chip memory area. The trace buffer stores trace information during capture that otherwise passes immediately through the trace port at the operating frequency of the core.

When capture is complete, the stored information can be read out at a reduced clock rate from the trace buffer using the JTAG port of the SoC, instead of through a dedicated trace port.

This is a two-step process that avoids implementation of a wide trace port that has many high-speed device pins. In effect, a zero-pin trace port is created where the device already has a JTAG port and associated pins.

(iv) Software access to trace buffer:

Buffered trace information can be accessed through an APB slave-based memory-mapped peripheral included as part of the trace buffer. Internal diagnostics can be performed on a closed system where a JTAG port is not normally brought out.

(v) Real-time debug facilities:

The ARM1176JZ-S processor contains an Embedded ICE-RT logic unit that provides the following real-time debug facilities:

- Up to six breakpoints.
- Thread-aware breakpoints.
- Up to two watchpoints.
- *Debug Communications Channel* (DCC).

The Embedded ICE-RT logic connects directly to the core and monitors the internal address and data buses. The Embedded ICE-RT logic can be accessed in one of two ways:

- Executing CP14 instructions.
- Through a JTAG-style interface and associated TAP controller.

The Embedded ICE-RT logic supports two modes of debug operation:

1. Halting debug-mode: On a debug event, such as a breakpoint or watchpoint, the debug logic stops the core and forces the core into Debug state. This enables the internal state of the core to examine and the external state of the system, independently from other system activity. When the debugging process completes, the core and system state is restored and normal program execution resumes.

2. Monitor debug-mode: On a debug event, the core generates a debug exception instead of entering Debug state, as in Halting debug-mode. The exception entry activates a debug monitor program that performs critical interrupt service routines to debug the processor. The debug monitor program communicates with the debug host over the DCC.

(vi) Debug and trace environment:

Several external hardware and software tools are available to enable real-time debugging using the Embedded ICE-RT logic and execution trace using the ETM.

8. Instruction Cycle Summary and Interlocks:

- The pipelined architecture of ARM9E-S overlaps the execution of several instructions in different pipeline stages.
- The instruction cycle count is the number of cycles that an instruction occupies the execute stage of the pipeline.
- The other pipeline stages (Fetch, Decode, Memory, Writeback) are only occupied for one cycle by any instruction (in this model, interlock cycles are grouped in with the instruction generating the data that creates the interlock condition, not the instruction dependent on the data).
- The request, address, and control signals on both the instruction and data interfaces are pipelined so that they are generated in the cycle before the one to which they apply.
- The instruction address, **IA[31:1]**, is incremented for prefetching instructions in most cases. The increment varies with the instruction length 4 bytes in ARM state or 2 bytes in Thumb state.
- The letter i is used to indicate the instruction length.

9. Vector Floating-Point (VFP) :

- The VFP supports floating point arithmetic operations. It is the part of ARM1176JZF-S.
- The VFP coprocessor is mapped as coprocessor numbers 10 and 11. Software can determine whether the VFP is present by the use of the Coprocessor Access Control Register.
- It supports single and double precision arithmetic on vector-vector, vector-scalar and scalar-scalar data sets. Vectors can consist of up to eight single precision or four double-precision elements.
- VFP has its own bank of 32 registers which can be used in pairs for double-precision operands or it can operate loads and stores of VFP registers in parallel with arithmetic operations.
- The VFP supports all five floating exceptions such as invalid operation, divide by zero, overflow, underflow, inexact defined by the IEEE 754 standard.
- These exception traps can be individually enabled or disabled.
- All rounding modes are supported and basic single/double formats are used.

10. System Control:

- The control of the memory system and its associated functionality, and other system-wide control attributes are managed through a dedicated system control coprocessor, CP15.
- The system control and configuration registers provide overall management of:
  - Memory functionality.
  - Interrupt behaviour.
  - Exception handling.
  - Program flow prediction.
  - Coprocessor access rights for CP0-CP13, including the VFP, CP10-11.
- The system control and configuration registers also provide the processor ID and information on configured options.
- The system control and configuration registers consist of 18 read-only registers and seven read/write register.
- Some of the functionality depends on how we set external signals at reset.

11. Interrupt Handling:

- Interrupt handling in the ARM1176JZ-S processor is compatible with previous ARM architectures, but has several additional features to improve interrupt performance for real-time applications.

The interrupt handling contains following sections:

(i) Vectored interrupt controller port

(ii) Low interrupt latency configuration

(iii) Configuration

(iv) Exception processing enhancements.

(i) Vectored interrupt controller port:

- The core has a dedicated port that enables an external interrupt controller such as the ARM *Vectored Interrupt Controller* (VIC) to supply a vector address along with an *interrupt request* (IRQ) signal. This provides faster interrupt entry but we can disable it for compatibility with earlier interrupt controllers.

(ii) Low interrupt latency configuration:

- This mode minimizes the worst-case interrupt latency of the processor with a small reduction in peak performance or instructions-per-cycle. The behaviour of the core can be tuned to suit the requirements of the application.
- The low interrupt latency configuration disables HUM operation of the cache. In low interrupt latency configuration, on receipt of an interrupt, the ARM1176JZ-S processor abandons any pending restartable memory operations or restarts memory operations on return from the interrupt.
- To obtain maximum benefit from the low interrupt latency configuration, software must only use multi-word load or store instructions that are fully restartable.
- The software must not use multi-word load or store instructions on memory locations that produce side-effects for the type of access concerned. This applies to ARM (LDC, all forms of LDM, LDRD and STC, and all forms of STM and STRD), Thumb (LDMIA, STMIA, PUSH and POP).
- To achieve optimum interrupt latency, memory locations accessed with these instructions must not have large number of wait-states associated with them.

To minimize the interrupt latency, the following is recommended:

- Multiple accesses to areas of memory marked as Device or Strongly Ordered, must not be performed.
- Access to slow areas of memory marked as Device or Strongly Ordered must not be performed. That is, those that take many cycles in generating a response.
- SWP operations must not be performed to slow areas of memory.

(iii) Configuration:

- The processor for low interrupt latency mode can be configured by use of the system control coprocessor. To ensure that a change between normal and low interrupt latency configurations is synchronized correctly. Software systems should be used so that only by changing the configuration, interrupts can be disabled.

(iv) Exception processing enhancements:

- The ARMv6 architecture contains several enhancements to exception processing, to reduce interrupt handler entry and exit time such as SRS to save return state to a specified stack frame, RFE to return from exception and CPS which will directly modify the CPSR.

2.7 CPU Pipelining Stages

- The pipeline consists of 3 stages:
  1. Fetch stage
  2. Decode stage
  3. Execute stage
- Below Fig. 2.8 shows the pipeline stages of the processor and the pipeline operations that take place at each stage.

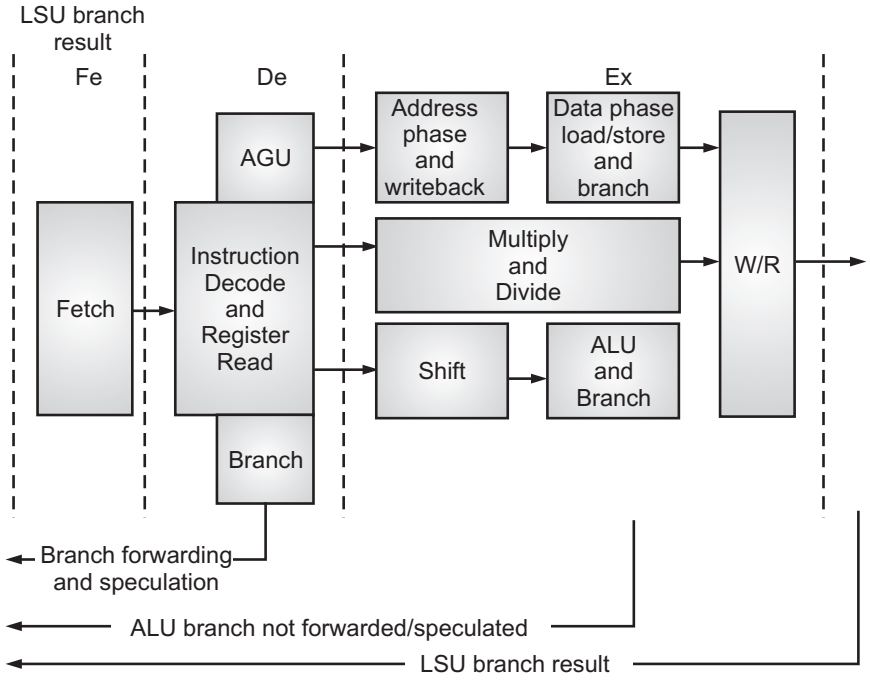


Fig. 2.8 : Cortex-M3 pipeline stages

- The names of the pipeline stages and their functions are:

Fe: Instruction fetch where data is returned from the instruction memory.

De: Instruction decode, generation of LSU address using forwarded register ports, and immediate offset or LR register branch forwarding.

Ex: Instruction execute, single pipeline with multi-cycle stalls, LSU address/data pipelining to AHB interface, multiply/divide, and ALU with branch result.

- The pipeline structure provides a pipelined 2-cycle memory access with no ALU usage penalty, address generation forwarding for pointer indirection.

Pipeline Stages:

- Below Fig. 2.9 shows pipeline stages such as the two Fetch stages, a Decode stage, an Issue stage and the four stages of the ARM1176JZ-S integer execution pipeline. These eight stages make up the processor pipeline.

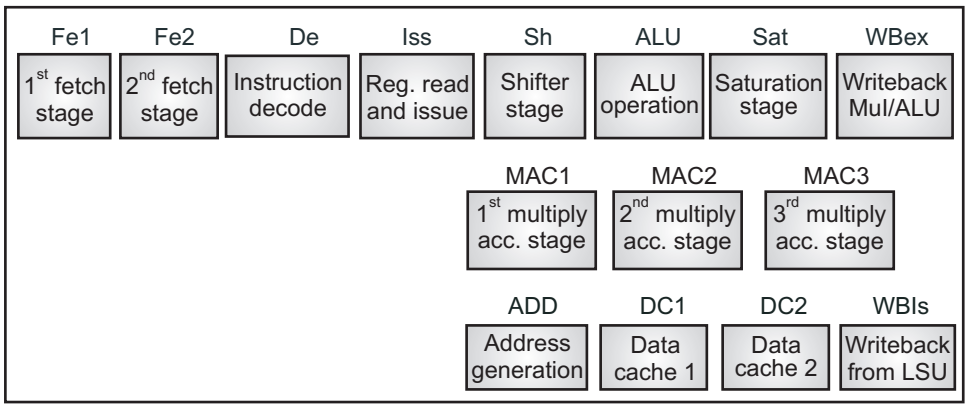


Fig. 2.9 : ARM1176JZ-S pipeline stages

- Above Fig. 2.9 shows the pipeline operations as:

Fe1: First stage of instruction fetch where address is issued to memory and data returns from memory.

Fe2: Second stage of instruction fetch and branch prediction.

De: Instruction decode.

Iss: Register read and instruction issue.

Sh: Shifter stage.

ALU: Main integer operation calculation.

Sat: Pipeline stage to enable saturation of integer results.

WBex: Write back of data from the multiply or main execution pipelines.

MAC1: First stage of the multiply-accumulate pipeline.

MAC2: Second stage of the multiply-accumulate pipeline.

MAC3: Third stage of the multiply-accumulate pipeline.

ADD: Address generation stage.

DC1: First stage of data cache access.

DC2: Second stage of data cache access.

WBls: Write back of data from the Load Store Unit.

- By overlapping the various stages of operation, the ARM1176JZ-S processor maximizes the clock rate achievable to execute each instruction. It delivers a throughput approaching one instruction for each cycle.
- The Fetch stages can hold up to four instructions, where branch prediction is performed on instructions ahead of execution of earlier instructions.
- The Issue and Decode stages can contain any instruction in parallel with a predicted branch.
- The Execute, Memory and Write stages can contain a predicted branch, an ALU or multiply instruction, a load/store multiple instruction and a coprocessor instruction in parallel execution.

Typical ALU Pipeline Operations:

- Below Fig. 2.10 shows all the operations in each of the pipeline stages in the ALU pipeline, the load/store pipeline, and the HUM buffers.

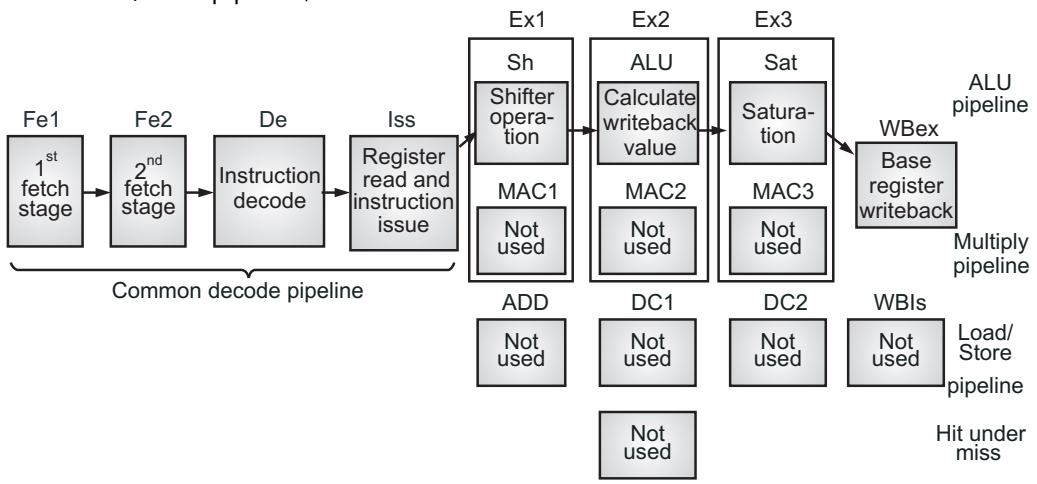


Fig. 2.10 : Typical ALU operations in pipeline stages

Typical ALU Operation:

- Below Fig. 2.10 shows a typical ALU operation and data processing instruction. The processor does not use the load/store pipeline or the HUM buffer.

Typical Multiply Operation:

- Below Fig. 2.11 shows a typical multiply operation. The MUL instruction can loop in the MAC1 stage until it has passed through the first part of the multiplier array enough times.
- The MUL instruction progresses to MAC2 and MAC3 where it passes through the second half of the array once to produce the final result.

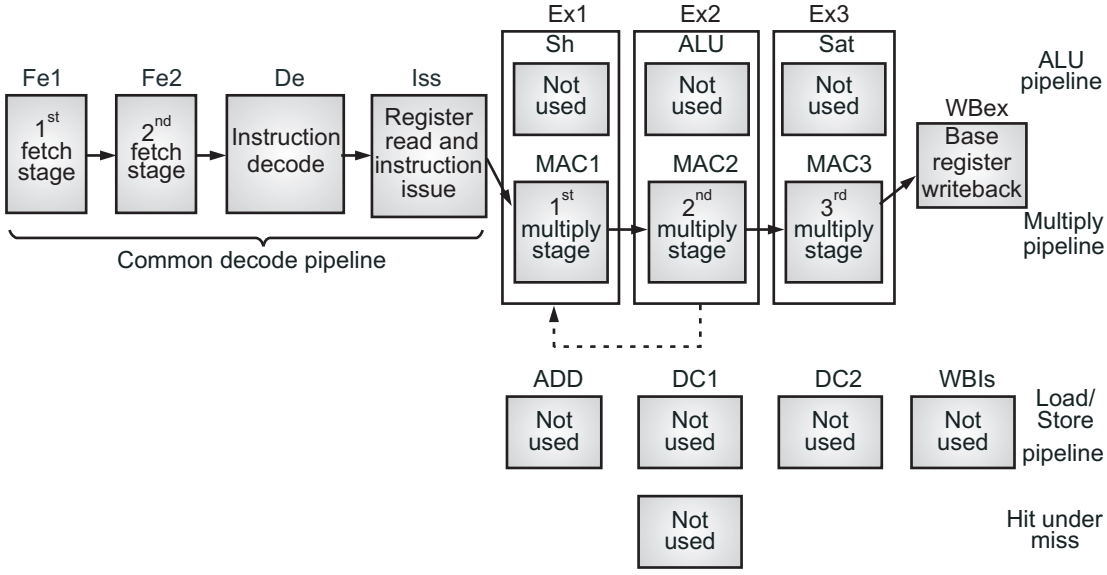


Fig. 2.11 : Typical multiply operation

- Below Fig. 2.12 shows pipeline flow for Load/Store operation.

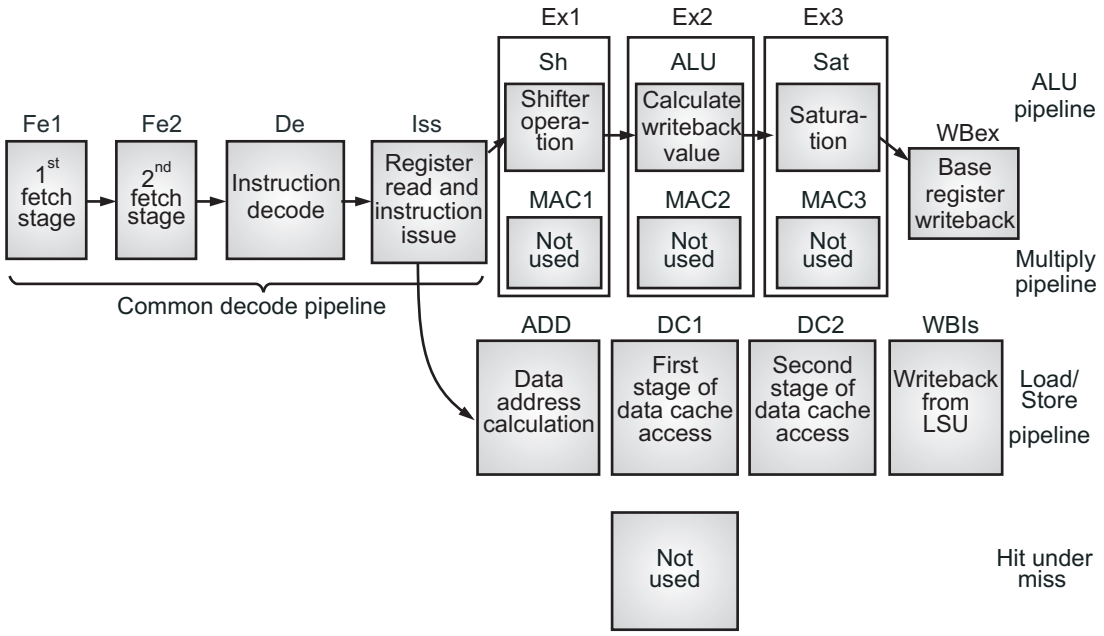


Fig. 2.12: Pipeline flow for load/store operation

Software Pipelining:

- Software pipelines is a parallel processing technique that enables multiple data elements to be processed simultaneously by breaking the computation into a series of sequential stages.
- Pipelines are common in both hardware and software. For example, application processors and GPUs use hardware pipelines. The graphics standard OpenGL ES is based on a virtual pipeline.
- In a pipeline, a complete process is divided into a series of stages. A data element is processed in one stage and the results are then passed to the next stage.
- Because of the sequential nature of a pipeline, only one stage is used at a time by a particular data element. This means that, the other stages can process other data elements.
- Software pipelines can be used in specific application to process different data elements.
- For example, a game requires many different operations to happen. A game might use a similar pipeline to this:
  1. The input is read from the player.
  2. The game logic computes the progress of the game.
  3. The scene objects are moved based on the results of the game logic.
  4. The physics engine computes positions of all objects in the scene.
  5. The game uses OpenGL ES to draw objects on the screen.

2.8 CPU Cache Organization

- Each cache is implementation-defined and can be one, two or four-way set associative cache of configurable size. They are physically indexed and physically addressed.
- The cache sizes are configurable with sizes in the range of 1 to 64 kB, but the maximum clock frequency might be affected if you increase the cache sizes beyond 16 kB.
- Fig. 2.13 shows blocked diagram of cache subsystem.
- Both the instruction cache and the data cache are capable of providing two words per cycle for all requesting sources.
- The cache way size can be varied between 1 kB and 16 kB in powers of 2. A 1 kB cache size must be implemented as a 1 way cache, and a 2 kB cache must be implemented as a 2 way cache. All other cache sizes must be implemented as 4 way set associative. The cache line length is fixed at eight words (32 bytes).
- The maximum cache way size that the processor supports is 16 kB. The minimum cache way size that the processor supports is 1 kB. Instruction cache and data cache together or instruction cache and data cache individually can be disabled.
- If a cache is implemented within the ARM1156T2F-S processor, way 0 must be present.
- Write operations must occur after the Tag RAM reads and associated address comparisons have completed.
- A three-entry Write Buffer is included in the cache to enable the written words to be held until they can be written to cache. One or two words can be written in a single store operation.
- The addresses of these outstanding writes provide an additional input into the Tag RAM comparison for reads.
- To avoid a critical path from the Tag RAM comparison to the enable signals for the data RAMs, there is a minimum of one cycle of latency between the determination of a hit to a particular way and the start of writing to the data RAM of that way. This requires the Cache Write Buffer to be able to hold three entries, for back-to-back writes.
- Accesses that read the dirty bits must also check the Cache Write Buffer for pending writes that result in dirty bits being set. The cache dirty bits for the data cache are updated when the Cache Write Buffer data is written to the RAM. This requires the dirty bits to be held as a separate storage array (significantly, the tag arrays cannot be written, because the arrays are not accessed during the data RAM writes), but permits the dirty bits to be implemented as a small RAM.
- The other main operations performed by the cache are cache line refills and write-back. These occur to particular cache ways, which are determined at the point of the detection of the cache miss by the victim selection logic.
- To reduce overall power consumption, the number of full cache reads is reduced by the sequential nature of many cache operations, especially on the instruction side.

- On a cache read that is sequential to the previous cache read, only the data RAM set that was previously read is accessed, if the read is within the same cache line.
- The Tag RAM is not accessed at all during this sequential operation.
- Cache line refills can take several cycles. The cache line length is of eight words.
- The control of the level one memory system and the associated functionality, together with other system wide control attributes are handled through the system control coprocessor, CP15.
- Below Fig. 2.13 shows the block diagram of the cache subsystem. This Fig. 2.13 does not show the cache refill paths.

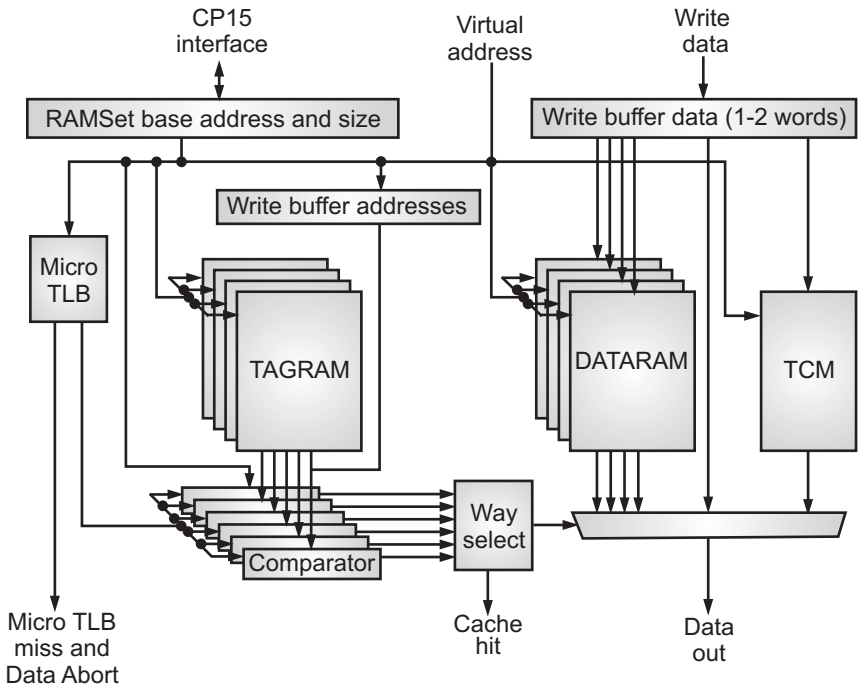


Fig. 2.13 : Level one cache block diagram

- Cache is Harvard implementation.
- Cache replacement policies are Pseudo-Random or Round Robin which is controlled by the RR bit in CP15 register c1.
- MicroTLB determines if cache lines are write-back or write-through.
- It contains both secure and non-secure data in cache lines.
- Processor handles branches first time execution when no history is available for dynamic prediction for the prefetch unit.
- Integer Core (IC) uses static branch prediction and return stack.

- Prefetch Unit (PU) uses dynamic branch prediction.
- When a branch is resolved, the PU receives information from the IC and either allocates space in the Branch Target Address Cache (BTAC) or updates an entry.
- Branches are resolved at or before the third execution stage.

2.9 Branch Prediction

- Branch prediction uses both static and dynamic techniques. Dynamic branch prediction is used by default. But when there is no information history, static prediction is used instead.
- Branch prediction predicts:
  - That there is a branch instruction at a given address.
  - The type of the branch:
    (i) Unconditional or conditional.
    (ii) Immediate or load.
    (ii) Normal branch, function call, or function return.
  - The target address or the state of the branch, either ARM or Thumb.
  - The direction of conditional branch, either taken or not taken.
  - There are two branch prediction methods:
    (i) Static branch prediction.
    (ii) Dynamic branch prediction.

(i) Static branch prediction:

- The static branch prediction is based on decoding the instruction. Therefore, it can see branches on fresh code without any history, but the prediction is done only at the decoding stage. So no fetch decision can be made before this stage, that is, speculative fetches from the branch target cannot be made.
- Static branch prediction is based on the characteristics of the branch instruction.
- It uses no history information.
- ARM1176JZF-S predicts all forward conditional branches not taken and all backward branches taken.
- Added to mitigate the trouble experienced by the miss when first encountering the branch by the predictor.

(ii) Dynamic branch prediction:

- The dynamic branch prediction estimates the instructions based on history, so that it can fetch speculatively to an arbitrary chosen branch of the execution code.
- More hardware is required, but it saves some unnecessary i-cache lookup/memory accesses and the prediction quality is higher for previously seen branches.

- By default, the dynamic branch prediction is used and if there is no information in its history, the static prediction is used instead.
- Many branch instructions are conditional. For conditional branches, whether the branch should be taken cannot be determined until the instruction is executed.
- The processor makes a prediction about whether the branch will be taken and fetches based on the prediction.
- The processor must also be able to detect when it gets the prediction wrong and re-fetch from the correct location.
- Branch prediction logic is an important factor in achieving high throughput in Cortex-R series processors.
- If branch prediction is not specified, one has to wait until a conditional branch executes before you could determine where to fetch the next instruction from.
- It uses a Branch Target Address Cache (BTAC) as the first line of branch prediction that hold virtual target addresses.
- Prediction history of a branch is stored as a two-bit value in the BTAC. BTAC is a 128-entry direct-mapped cache structure.
- Two bit values represent the following four states: Strong predict branch taken, Weak predict branch taken, Strong predict branch not taken and Weak predict branch not taken.

2.10 Branch Folding

- Branch folding is a technique where the Branch instruction is removed from the pipeline and is stored in a buffer, which is executed on all dynamic predicted branches.
- It can improve the Branch CPI to under 1.
- Predicted branches that lead directly to another branch.
- Branches that are have been cancelled when fetched.

Branch instructions are removed from the pipeline if the following conditions are met:

- The instruction is not a branch with a link (address is stored in a linked register).
- The instruction does not point to a code sequence that contains a branch in the first two instructions.
- The instruction is not break-pointed.
- The instruction is not aborted. This method can produce a CPI for branch instructions that is much lower than 1.

2.11 GPU Overview

- The GPO is specially designed to speed up the operation of image calculations.
- Broadcom Videocore IV uses OpenGL ES2.0, Performance: 24 GFLOPS, RPi can play 1080p, Blu-Ray quality videos, Graphical capabilities are similar to the those of the original XBOX.

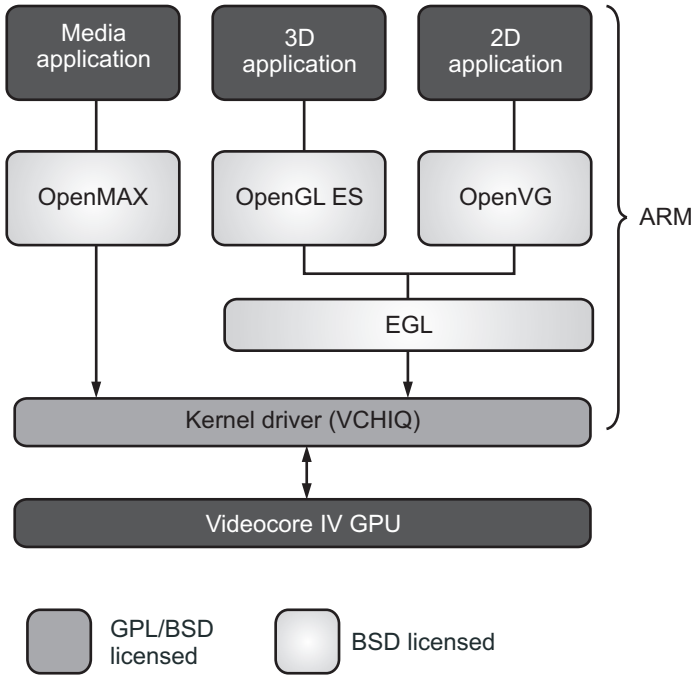


Fig. 2.14 : Raspberry Pi software architecture

- BroadCom VideoCore 4 GPU is shown in above Fig. 2.14. It contains Tile-based render (TBR) which uses 4 cores. It has 40 nm technology.
- It is capable of Blu-Ray quality of 1080p with H.264 at 40Mb/s.
- Its graphics performance is similar to the Xbox 1.
- It has texture filtering and DMA infrastructure.
- It has 24 GFLOPS of general purpose computational power.
- It has OpenGL ES 1.1, OpenGL ES 2.0, hardware accelerated OpenVG 1.1, Open EGL and OpenMax.

2.12 Software

- Raspbian has been the most popular among the operating systems due to the boost provided by the Raspberry Pi foundation.
- It is a Debian based Linux operating system easily available at the foundation website. Some of the popular operating systems are:
  - Raspbian
  - CentOS
  - Fedora
  - Ubuntu MATE
  - Kali Linux
  - Ubuntu Core
  - Windows 10 IoT Core

- RISC OS
- Slackware
- Debian
- Arch Linux ARM
- Android Things
- SUSE
- FreeBSD
- NetBSD

Think Over It

- Is the iPhone an example of SoC? Is the system saved on a processor in case of iphone?
- Which fabrication technologies are used in SoC?
- Which technology is used in Videon?
- What is the difference between SoC and CPU? 'Soc will replace CPU in coming years', comment.
- What are Nolocal and Global variables used in Python?

Points to Remember

- SoC is a complete electronic system which may contain analog, digital or radio frequency functions.
- The basic purpose of using SoC is to have all components on a single chip with minimum components.
- If any component on SoC is not functioning properly, then it cannot be replaced. In that case, entire SoC has to be replaced.
- Raspberry Pi is a small single board computer (SBC).
- Raspberry Pi is credit size computer which can be plugged into a monitor. It acts as minicomputer by connecting keyboard, mouse, and display.
- Raspberry Pi uses Broadcom BCM2835 System on Chip which is ARM processor and Videocore Graphics Processing Unit (GPU). It is the heart of the Raspberry Pi which controls the operations of all the connected devices and handles all the required computations.
- The first generation of Raspberry Pi used BroadCom2835 SoC. The Raspberry Pi 2 used BCM2836. The third generation of Raspberry Pi used BCM2837 SoC.
- The BroadCom2835 is a System on Chip (SoC). It contains multimedia capabilities which are used in mobile phones and portable devices.
- The ARM1176JZ-S processor is an integer core which implements ARM11. It supports Java bytecodes.

- The **main components** of the **ARM1176JZ-S** processor are :
  - Integer core
  - Load store unit (LSU)
  - Prefetch unit
  - Memory system
  - AMBA AXI interface
  - Coprocessor interface
  - Debug
  - Instruction cycle summary and interlocks
  - System control
  - Interrupt handling
- The main instruction set categories are:
  - Branch instructions
  - Data Processing instructions
  - Status register transfer instructions
  - Load and store instructions
  - Coprocessor instructions
  - Exception-generating instructions.
- The Integer Core uses both static and dynamic branch prediction.
- The main purpose of Prefetch Unit (PFU) is to perform speculative fetch of instructions ahead of the DPU by predicting the outcome of branch instructions and to format instruction data in a way that aids the DPU in efficient execution.
- The *Memory Management Unit* (MMU) has a unified *Translation Lookaside Buffer* (TLB) for both instructions and data.
- The ARM1176JZ-S processor implements the *Fast Context Switch Extension* (FCSE) and high vectors extension that are required to run Microsoft Windows CE.
- The MMU is responsible for protection checking, address translation, and memory attributes, and some of these can be passed to an external level two memory system.
- The memory translations are cached in MicroTLBs for each of the instruction and data caches, with a single Main TLB backing the MicroTLBs.
- The Advanced Microcontroller Bus Architecture (AMBA) protocols are an open standard, on-chip interconnect specification for the connection and management of functional blocks in a SoC.
- The pipeline consists of 3 stages:
  - Fetch stage
  - Decode stage
  - Execute stage

- Cache is Harvard implementation.
- Cache replacement policies are Pseudo-Random or Round Robin which is controlled by the RR bit in CP15 register c1.
- MicroTLB determines if cache lines are write-back or write-through.
- Integer Core (IC) uses static branch prediction and return stack.
- Prefetch Unit (PU) uses dynamic branch prediction.
- Static branch prediction is based on the characteristics of the branch instruction. It uses no history information.
- The dynamic branch prediction estimates the instructions based on history, so that it can fetch speculatively to an arbitrary chosen branch of the execution code.
- Branch folding is a technique where the Branch instruction is removed from the pipeline and is stored in a buffer, which is executed on all dynamic predicted branches.
- Raspbian has been the most popular among the operating systems due to the boost provided by the Raspberry Pi foundation.

Exercises

[A] True or False :

1. A System-on-chip (SoC) is an integrated circuits (IC) which has most of the components of a computer or any electronic system.
2. SoC consumes more power.
3. Raspberry Pi is credit size computer which can be plugged into a monitor.
4. High Definition Multimedia Interface is used for transmitting video or digital audio data to computer monitor or to digital TV.
5. GPIO pins of Raspberry works on 5V.

[B] Fill in the Blanks :

1. SoC has ____, ____, ____ and ____.
2. SoC consumes _____ power.
3. Raspberry Pi is a type of _____ .
4. _____is an official OS provided by Raspberry Pi foundation.
5. HDMI stands for ____ _____ _____ _____
6. Raspberry Pi is a ____ pin SBC.
7. Raspberry Pi uses _____ processor.

[C] Multiple Choice Questions :

1. Raspberry Pi provides on-chip _____ modules.
   (a) UART (b) I2C
   (c) I2S (d) All of these

2. The MMU is responsible for _____ .
   (a) protection checking (b) address translation
   (c) memory attributes (d) all of these
3. The AMBA protocol stands for _____ .
   (a) Advanced Microcontroller Bus Architecture
   (b) Advanced Microprocessor Bus Architecture
   (c) Arithmetic Microcontroller Bus Architecture
   (d) None of these
4. The ARM1176JZ-S processor contains an Embedded ICE-RT logic unit that provides _____ .
   (a) up to 6 break points (b) thread-aware breakpoints
   (c) up to 2 watch points (d) All of these
5. CPU pipelining stages contains _____ .
   (a) fetch stage (b) decode stage
   (c) execute stage (d) All of these
6. _____ is the official OS provided by Raspberry Pi foundation.
   (a) Raspbian (b) Debian
   (c) Linux (d) None of these

[D] Short Answer Questions :

1. What is SoC?
2. Write any one advantage and disadvantage of SoC.
3. What are the main blocks of SoC?
4. What is Raspberry Pi?
5. Write main blocks of Raspberry Pi.
6. Which processor is used in Raspberry Pi?
7. Write long form of HDMI and GPIO.
8. Write function of GPIO pins available in Raspberry Pi.
9. List various models of Raspberry Pi available in market today.
10. Write any two features of Raspberry Pi.
11. List Peripherals used in BCM2835.
12. Write various blocks of BroadCom2835.
13. List main components of the ARM1176JZ-S processor.
14. What are the different types of branch prediction?
15. Write any two features of DMA.
16. What is MMU?

17. What do you mean by AMBA AXI interface?
18. Write CPU pipelining stages.
19. What is the function of MicroTLB?
20. What is branch folding?
21. List blocks of GPU used in Raspberry Pi.
22. Which software is most popular for Raspberry Pi?

[E] Long Answer Questions :

1. What is SoC? Write its advantages and disadvantages.
2. Differentiate between SoC and SBC.
3. Draw and explain the architecture of SoC with the help of neat diagram.
4. Write a short note on Raspberry Pi.
5. List features of Raspberry Pi.
6. Draw and explain the architecture of Raspberry Pi with the help of neat diagram.
7. Write the functions of following:
   GPIO, HDMI, SD card, Ethernet, Camera module.
8. Write a short note of comparison of Raspberry Pi modules.
9. Compare Raspberry Pi Models with reference to SoC, CPU, RAM, GPU, storage.
10. Write architectural features of Raspberry Pi.
11. Write a short note on peripherals used in BCM2835.
12. Draw block diagram of BroadCom2835 and explain any three blocks.
13. Draw the diagram of CPU architecture of ARM116JZ-S.
14. Write features of ARM1176JZ-S.
15. Draw the diagram of ARM1176JZ-S processor and list main components of it.
16. Write a short note on:
    (i) Integer core
    (ii) Load store unit (LSU)
    (iii) Prefetch unit
    (iv) Memory system
    (v) AMBA AXI interface
    (vi) Coprocessor interface
    (vii) Debug
    (viii) Instruction cycle summary and interlocks
    (ix) System control
    (x) Interrupt handling

17. Explain CPU pipelining stages.
18. Explain Cortex-M3 pipeline stages with the help of neat diagram.
19. Explain ARM1176JZ-S pipeline stages with the help of neat diagram.
20. Draw the diagram of Typical ALU operations in pipeline stages.
21. Draw the diagram of Typical multiply operation.
22. Draw the diagram of Pipeline flow for Load/Store operation.
23. What is software pipelining?
24. Explain CPU cache organization.
25. Draw and explain level one cache block diagram.
26. Write a short note on branch prediction.
27. Explain the concept of branch folding.
28. Write a short note on GPU used in Raspberry Pi.
29. List various softwares used for Raspberry Pi.
30. Write a short note on pipelining of Raspberry Pi.

Answers

[A] True or False :

(1) True (2) False (3) True
(4) True (5) False

[B] Fill in the Blanks :

(1) CPU, GPU, Memory and I/O devices (2) Less (3) SBC
(4) Raspbian (5) High Definition Multimedia Interface (6) 40
(7) ARM1176JZF-S

[C] Multiple Choice Questions :

1. (d)	2. (d)	3. (a)	4. (d)	5. (d)	6. (a)

Unit 3...

Programming Using Python

Guido van Rossum

Guido van Rossum a Dutch programmer was born on 31st January 1956. He is known as the creator of the Python programming language, for which he was the "Benevolent dictator for life" (BDFL) until he stepped down from the position in July 2018. He is currently a member of the Python Steering Council, however he has withdrawn from nominations for the 2020 Python Steering Council. Van Rossum was born and raised in the Netherlands where he received a master's degree in mathematics and computer science from the University of Amsterdam in 1982. He has a brother, Just van Rossum, who is a type designer and programmer who designed the typeface used in the "Python Powered" logo.

Introduction

- In last chapter, Raspberry Pi and its architecture, the processor used in Raspberry Pi and pipelining stages were discussed.
- This chapter deals with Operating System used for Raspberry Pi, its installation, and various types of Operating Systems.
- This chapter will also give details of Python programming, data types, variables used in Python, conditional statements, functions and few arithmetic programs.

3.1 Overview of Raspbian OS (Operating System)

- Raspberry Pi OS which is formerly known as Raspbian is a Debian-based operating system for Raspberry Pi. As Raspberry Pi Foundation has recommended Raspbian as the primary operating system for Raspberry Pi family, it is widely used for compact single board computers of Raspberry Pi from 2015.
- Raspberry Pi OS is a free operating system based on Debian which is optimized for the Raspberry Pi hardware.
- Raspberry Pi OS comes with over 35,000 packages which is precompiled software bundled in a nice format for easy installation on Raspberry Pi.
- Raspberry Pi OS is a community project under active development, with an emphasis on improving the stability and performance of as many Debian packages as possible.
- Raspbian is a competent and versatile operating system which is suitable for a PC and it provides a command line, a browser and tons of other programs.

- Raspberry Pi as running Raspbian can be used as a cheap and effective home computer, or it can be used as a springboard. It can be used in countless other functional devices, from wireless points to retro gaming machines.

3.2 Installation of Raspbian

- Raspbian is the most popular operating system for Raspberry Pi. The installation process of Raspbian is same for all models of Raspberry Pi.
- In Raspbian, thousands of pre-built libraries to perform many tasks and optimize the OS are available.
- Installing Raspbian on Raspberry Pi is very straightforward. Raspbian software has to be downloaded.
- MicroSD card, a computer slot, Raspberry Pi board with basic peripherals (a mouse, keyboard, screen and power supply) are required.
- Raspbian can be installed using NOOBS which is an operating system installation manger.

Steps for installing Raspbian are described as below:

Step 1: Downlaod Raspbian:

Download latest version of Raspbian.

Step 2: Unzip the file:

Raspbian OS is compressed, so files should be unzipped. Depending on built-in utilities, certain programs are used to unzip the files. For Windows users, 7-zip is used. For Mac users, The Unarchiver and for Linux, Unzip is used to unzip the files.

Step 3: Write the disc image on microSD card:

Downloaded OS require image writer to write into the SD card (microSD card). So download 'win32 disk imager' and write the disc image to it. For Windows users, 'Win32 Disk Imager' is required. For Mac users, disk utility is required and for Linux users, Etcher is usually used.

The process of writing image is slightly different for different users. For each of these programs, destination has to be selected to make sure that microSD card chosen and disc image that is the unzipped Raspbian file is selected. Then select double-check and hit the button to write.

Step 4: Put the microSD card in Raspberry Pi and boot up:

Once the disc image has been written to the microSD card, put it in Raspberry Pi board, plug in the peripherals and provide power supply to it. Raspbian will boot directly to the desktop. The default credentials for username are *pi* and password id *raspberry*.

- After completing above steps, Setup Options window will open, on which few settings have to be made.
- Below steps have to be followed:
  1. Select first option in the list of the setup options window which is 'Expand File system' option and press enter key. This will make use of all the space present on the SD card as a full partition. It will expand the OS to fit the whole space on the SD card which can be used as the storage memory for the Raspberry Pi.

2. Select third option in the list of the setup option menu as 'Enable Boot To Desktop/Scratch' window option. The window looks like as below (Fig. 3.1).

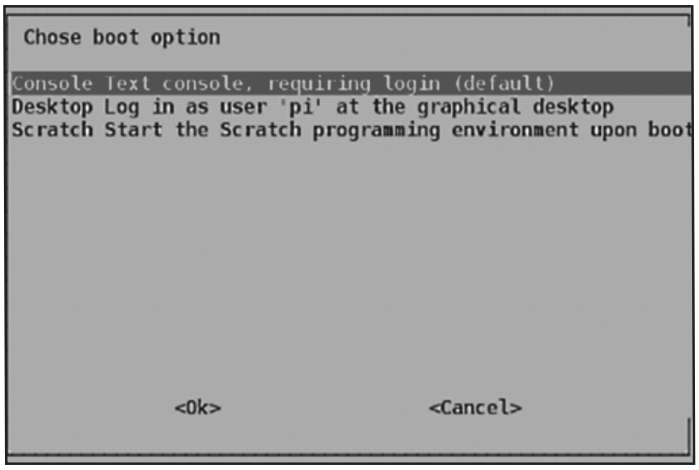


Fig. 3.1

- After completing above steps, it may ask to reboot the Raspberry Pi then every time system has to be rebooted.
- After reboot, if 'Setup Options' screen is not visible then follow below command:
  - In the 'choose boot option window', select the second option as Desktop Log in as user 'pi' at the graphical desktop and press the enter key. Once this is done, 'Setup Options' page will come, if not select the 'Ok' button at the bottom of this window which will go to previous window. This will boot into the desktop environment which is familiar screen. If this step is not done then the Raspberry Pi boots into a terminal each time with no GUI options.
  - Once above step is finished, select 'finish' button at the bottom of the page which will reboot automatically.

Step 5: Updating the firmware:

After rebooting, the desktop screen looks like the image below:

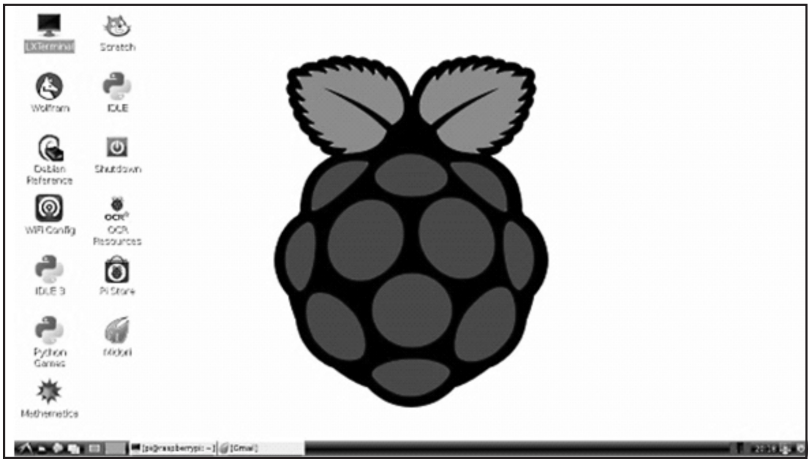


Fig. 3.2: Raspberry Pi screen image

Now, open a terminal and entre the command as:

sudo rpi-update

The Raspbian will get updated.

Updating the firmware is necessary because certain models of Raspberry Pi have bug or might not have all the required dependencies to run it smoothly. The latest firmware might have fix those bugs, so it is very important to update or upgrade the firmware.

3.3 Different Types of Operating Systems

- Raspberry Pi is the most popular SBC today. It is similar to computer and can perform many tasks.
- The functionality of Raspberry Pi can be extended by running OS on the device. It performs as a bridge between the user and Raspberry Pi hardware. OS helps in developing and executing the programs.
- OS enables the interaction between hardware and software. Also OS manages CPU, memory, disk drives, printers and all user interfaces.
- Raspbian is the official Raspberry Pi OS, however other OS are also available which can run on Raspberry Pi.

Most popular Raspberry Pi OS are listed below:

1. Raspbian:

- Raspbian is the official OS provided by Raspbery Pi foundation which can be used for all models of Raspberry Pi. Raspbian is free operation system which is known as the modified version of the popular OS Debian. It can serve all the general purposes required for Raspberry Pi.

Features of Raspbian:

- It can allow the user to set up a new password, username and interaction with wifi network.
- All applications having different specifications of Raspberry Pi models can run on Raspbian.
- The Software Tool of Raspbian allows user to download any required software from internet.
- It offers features like searching, toolbars configuration, key layout shortcuts, thumbnails and multi-page controlling with the newly introduced tool qpdf View.
- Preboot execution environment of Raspbian enabled users to boot their Raspberry using server through Ethernet.
- It overcomes the pi devices' security concern which is known as the pi/raspberry and eliminates the hassle to go online.

2. DietPi:

- DietPi is an extremely light Dibain OS. It is 3 times lighter than the other Raspberry Pi OS which are available today. Its installation is easy and automated. If dietpi.txt is configured before powering up, no user input will be required.

Features of DietPi:

- Diet Processing Tool which comes in package helps in determining the installed programs' priority level and control schedulers.
- With the help of customization ability, performance of the hardware and software can be adjusted.
- It uses minimum space in RAM, its size starts at just 400MB.
- It introduces lightweight Whiptail menus that take less time to execute the command line.
- It is suitable for new users and its performance can be bossted by controlling the amount of logging according to the need using DietPi-Ramlog.
- Installation, updating is made available automatically in DietPi OS. When there is an update available, no need to write an image for updating the system.

3. LIBREELC:

- LIBREELC is a small open source JEOS. The boot time is faster as compared with OpenELEC. It offers backbones for backdated hardware. It was launched in 2016 to generate better multimedia output than OpenELEC.

Features of LIBREELC OS:

- This OS can be installed in less than 20 minutes. It contains clear instruction, an SD creator app and an installer.
- It boots up within few seconds.
- With the help of Kodi, it offers standard customizations. It contains bunch of add-ons.
- It is managed by a team of developers and provide updates every month.
- It works closely with Kodi and regularly patches while maintaining security.

4. OSMC:

- It works best to manage media content. It is an open source software that has interface with many features. It is based on KodiOs which provides support virtually to any media content.

Features of OSMC:

- It has several customizable built-in-images that allow Raspberry to produce high-quality videos, pictures and other media contents.
- It takes very less time to complete the installation.
- This OS updates automatically. It will update itself automatically once every month.
- It offers Google play store. User can download, install, and start using the required applications whenever required.

5. RISC OS:

- RISC OS is the best Raspberry OS as it is intended to serve ARM processors. It enhances the performance and efficiency of the system.

Features of RISC OS:

- Designed by the original inventor of ARM. It is a unique open-source operating system for Raspberry Pi.

- It is maintained by a dedicated team of volunteers.
- It is not suitable for new users.
- It will take time to get used to the environment offered by RISC.
- It is a single-user OS and operates co-operative multitasking known as CMT. It is secured from OS corruption and boot time is very less.

6. Windows IoT Core:

- This is a powerful Raspberry Pi OS designed specially for writing sophisticated programs and making prototypes. It was intended to serve the developers and programmers. It has enabled the coders to make IoT projects using Raspberry Pi and Windows 10.

Features of Windows IoT Core OS:

- It focuses on security, connectivity, project development and integration with the cloud.
- It requires Windows 10 for running on any device.
- It contains Microsoft Visual Studio to work with any applications.
- It only allows a single Universal Windows Program application and background processes.
- It is compatible with ARM and IoT core can be used on SBCs like Raspberry Pi.
- IoT core can be blended with sensors like cameras, PIR sensors, servos and temperature sensors to extend the usability.

7. Lakka:

- Lakka is more suitable to develop computer games or even play games on a single board computer. It can turn Raspberry into a gaming console without keyboard or mouse. Also it contains a good user interface and a handful of customization features.

Features of Lakka OS:

- It can be installed on SD card easily, or it can be used as running live.
- The OS is free and lightweight.
- It has a dedicated setup of hardware.
- It handles and processes computing power for running games using a libretro core.
- It allows the user to connect several USB joypads.
- It consumes less power and runs smoothly on Raspberry and other low-end hardware architectures.
- It is developed and backed by a group of developers, designers and gamers.

8. RaspBSD:

- It is an open-source software which comes from FreeBSD 11. The developers have preconfigured it in two images for use in Raspberry Pi. The Berkeley Software Distribution has invented it and now it is one of the widely used single board OS.

Features of RaspBSD OS:

- This OS is very lightweight and mostly used as game consoles.
- It is intended to help newcomers to build projects using Raspberry Pie.

- It is regularly updated and uses only publicly available tools.
- It comes with Openbox and the LXDE graphical desktop, alongside FreeBSD package repositories which will be preconfigured.

9. RetroPie:

- It is built based on the Debian software library. It is suitable for emulating retro games on a single-board computer like Raspberry, ODroid C1/C2, or even PC. It provides a modern and user-friendly interface.

Features of RetroPie:

- RetroPie uses the Emulation Station front end and SBC to generate the best retro gaming experience for its users.
- It comes with a special preloaded SD card image that can boot the OS without facing any trouble.
- It is one of the very few OS that can be operated by installing on top of another operating system.
- IoT devices, running media-player and more than 50 other applications come pre-installed in this OS.

10. Ubuntu Core:

- Ubuntu is one of the widely used operating systems. This OS is designed for building and managing Internet of Things applications. It is open source and backed by many developers.

Features of Ubuntu Core OS:

- Ubuntu has 20+ other derivatives. So using this OS, user will be a member of an active and welcoming forum.
- It covers the basic sets of the platform, services and technologies to work more efficiently with IoT projects.
- This OS is lightweight and highly secure.
- Focuses on meeting the requirement of IoT devices and their distributors.
- Public and Private key is generated while two steps validation and authentication at every step makes it more secure.

11. Linutop:

- This OS is specially developed to start an Internet stall or digital marketing platform. It is based on Raspbian OS and is dedicated to rendering Web Kiosk or digital signage. Linutop is a small, lightweight OS and also comes with a hardware setup.

Features of Linutop OS:

- This OS can be easily used on Raspberry Pi B, B+, and get a smooth performance.
- It is more suitable for running businesses such as hotels, restaurants, shops, city halls, offices and museums.
- Linutop is a customized version of the same Linux distribution used by Xubuntu and Ubuntu/XFCE.

- It is capable of running multimedia software and emphasizes secure web browsing on single board computers.

12. Kali Linux:

- Kali is based on Linux specially designed for Raspberry Pi. It can be used on desktop computers. Additional tools can be used which are available on the website to extend the capabilities of certain features.

Features of Kali Linux OS:

- Class 10 SD card with at least 8 GB of data storage is required for installing a prebuilt Kali Linux image on Raspberry.
- This OS offers a lot of security and forensics tools to ensure the project or applications' security.
- Security can be ensured through research, testing, forensic reports, or even reverse engineering to accomplish the goal.
- The Kali is best for high computing projects.
- Kali can provide support to Ethical Hacking, it helps in Cracking Wi-Fi password, spoofing and testing networks.

13. Ubuntu Mate:

- Ubuntu is very popular today. Ubuntu Mate is specially designed and dedicated to run on Raspberry Pi. It is a good choice for users looking for an alternative to Raspbian.

Features of Ubuntu Mate OS:

- It is a Debian based Linux distribution which gets updates much faster.
- It has colourful interface and it offers Raspbian, Minecraft Pi or Scratch.
- The Ubuntu team maintains the kernel and updates can be done by expanding the file system automatically.
- It comes with HDMI, WIFI, Bluetooth, and Ethernet support. It can encode and decode the hardware-assisted video using FFmpeg.

14. OpenMedia Vault:

- It is Debian based OS which is intended to improve the network performance. It is suitable for network drive or storage applications running in Raspberry Pi.

Features of OpenMedia Vault OS:

- The OS can be updated using the Debian Package Management.
- It is more suitable to use in any office or home. Its performance can be improved by installing plugins from a vast collection.
- It supports traditional services of Raspberry Pi.
- It comes with Link aggregation, Wake On Lan, and IPv6 support.

15. Gentoo:

- This OS is very flexible. It is from Linux distribution and it is very lightweight. All packages and services are available with this OS. It is supported by Arch Linux, which is widely recognized for Raspberry Pi development.

Features of Gentoo OS:

- This OS can be used easily on Raspberry Pi. More functionality can be added using emerge and portage.
- It compiles new applications locally on the computer.
- After downloading new software, Source code can be extracted and developed according to user's preference.
- It is supported by onboard Raspberry Pi Ethernet, WiFi, and Bluetooth adaptors.

16. Kano:

- It is usually referred to as an entirely planned educational project and designed for the children. Kano manufactures computer kits to inspire children to learn how a computer works, how to write code, or how to work with basic projects. Children and the individuals who are interested in developing art, music, apps and games software can start with the starting kit distributed by Kano.

Features of Kano OS:

- Kano offers an open-source OS to use in Raspberry Pi and setup wizard guides the user after completing the installation.
- It contains several story modes and a fresh set of features. User need to create an account and set a username to use this.
- With Other applications like Minecraft, Youtube, web browsers are also available with this OS. These most used applications are usually located on the menu.
- User can start building small projects after installing the OS with the dedicated apps.
- This is a new OS. They have provided many books, resources and instruction videos on their website.

17. Rokos:

- Rokos is best for the projects related to cryptography. As the necessity to transform computer into a minor node is increasing, SBCs are becoming more popular.

Featuers of Rokos OS:

- This OS offers solution to start crypto mining for the people who cannot afford high-end computers or hardware.
- It provides support to users to create Bitcoin wallet or portfolio.
- This OS helps the user to earn and educate them.
- It comes with fully functional Bitcoin and OKcash node. The OpenBazar server is also involved with the package.
- It focuses on updating security, regulations and optimizing the system regularly. Dependencies for compilation are also added.

18. Minibian:

- All the features and applications are packed into this minimalistic version of Raspbian. As it is developed and managed by the same team, it takes advantage of Raspbian. It focuses on eliminating the unneeded apps to improve overall performance, although it does not offer a graphical user interface.

Features of Minibian OS:

- The environment of the Minibian is more stable and it can provide a lightweight system for the embedded system.
- The same repositories and binary projects like Raspbian can be used.
- It is not suitable for advanced projects and less popular among the developers.
- Boot ability through network or ethernet is now available.

19. Chromium OS:

- Google offers this open-source version of the Chromium OS. That was intended to use on Chromebook computers, but it is also available for Raspberry Pi. It can single-handedly convert the Raspberry Pi into a desktop PC as it allows users to run powerful applications using cloud computing rather than depending on the hardware resources.

Features of Chromium OS:

- It is more suitable for web browsing.
- It comes with all the applications offered by Google like Gmail, drive, access, docs, keeps etc.
- Chromium can be used on Raspberry Pi 3 or 3B+ devices as there are no images available for Pi Zero or Raspberry 4.

20. Alpine Linux:

- It is intended to serve the users who need more computational power. It is an independent Linux distribution that focuses on ensuring security, improving the system's efficiency and making the interface simple. It is more suitable for a network project or work with VPN, Routers and Firewalls.

Features of Alpine Linux:

- It is lightweight system and has an image for Raspberry Pi which will not occupy more than 50 MB of the storage.
- Any system according to the user's demand can be built and any application can be configured.
- Its own APK management tool can be used for upgrading the apps.
- It can be used as a workstation for XFCE, Firefox and Linux software.
- It provides a noise-free Linux environment.

- As discussed above, there are number of operating systems available for Raspberry Pi. User can select any one depending on the application.
- Today Raspberry Pi is the most demanding and attractive SBC.
- Raspberry Pi helps to build any prototypes and develop applications or software.
- Raspberry Pi can generate output like a desktop computer and has ability to serve individuals and small business.
- User should select the right OS to accomplish his goal.

3.4 Basic Python Programming (Script Programming)

- Python is becoming very popular nowadays. Python is a general purpose interpreted, interactive, object-oriented and high-level programming language.
- It was created by Guido van Rossum during 1985-1990 at the National Research Institute for Mathematics and Computer Science in the Netherlands.
- Python is copyrighted. Python source code is also available under the GNU General Public License (GPL).
- Python is derived from many other languages such as ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.
- Python is maintained by a core development team at the institute.
- Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation and it has fewer syntactical constructions than other languages.
- Today, Python is very essential to become a good software Engineer.

3.4.1 Features of Python

- Python is Interpreted, that means, Python is processed at runtime by the interpreter. Compiling a program is not required before executing it.
- Python is Interactive that means user can interact with the interpreter directly while writing the programs on Python prompt.
- Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is very easy to understand so it is known as beginner's language. It supports the development of a wide range of applications from simple text processing to WWW browsers to games.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It supports automatic garbage collection.
- Python is easy to read, easy to maintain. It has broad standard library.
- It is portable, extendable.
- It supports GUI programming.

3.4.2 Python Installation

- The documentation of Python is available on heet://www.python.org/doc/. The documentation is available in HTML, PDF and PostScript formats.
- Python is available for a wide variety of platforms. Only the binary code application for selected platform has to be downloaded and Python is installed.
- If the binary code for selected platform is not available then a C compiler has to be downloaded to compile the source code manually. Compiling the source code offers more flexibility in terms of choice of features.

The steps for installing Python on various platforms are given below:

(I) Unix and Linux Installation:

- Open a Web browser and go to https://www.python.org/ downloads/.
- Follow the link to download zipped source code available for Unix/Linux.
- Download and extract files.
- To customize some options, edit the Modules/Setup file.
- run ./configure script.
- make
- make install

This installs Python at standard location */usr/local/bin* and its libraries at */usr/local/lib/pythonXX* where XX is the version of Python.

(II) Windows Installation:

- Open a Web browser and go to https://www.python.org/ downloads/.
- Follow the link for the Windows installer python-XYZ.msi file where XYZ is the version to install.
- To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file on local machine and then run it and check whether the machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, accept the default settings, wait until the install is finished.

(III) Macintosh Installation:

- Get instructions, for the current version with extra tools to support development on the Mac from http://www.python.org/ download/mac/.
- For older Mac OS's before Mac OS X 10.3 (released in 2003), MacPython is available.
- Jack Jansen maintains it and access to the entire documentation is available at his website – http://www.cwi.nl/~jack/ macpython.html. Complete installation details for Mac OS installation are available on this site.

3.4.3 Python Variable

- Variables are nothing but reserved memory locations to store the values. When a variable is created, some space is reserved.
- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.
- By assigning different data types to variables, integers, decimals or characters in these variables can be stored.
- Python is completely object oriented. Variables or their types need not to be declared before using them. Every variable in Python is an object.
- Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when a value to a variable is assigned. The equal sign (=) is used to assign values to variables.

- For example :

```
num = 50            #num is of type int
str = "Chaitanya"  #str is of type string
```

- Variable name is known as identifier. There are few rules that have to follow while naming the variables in Python as given below:
  - The name of the variable must always start with either a letter or an underscore (_). For example: _str, str, num, _num are all valid names for the variables.
  - The name of the variable cannot start with a number. For example: 9num is not a valid variable name.
  - The name of the variable cannot have special characters such as %, $, # etc., they can only have alphanumeric characters and underscore (A to Z, a to z, 0-9).
  - Variable name is case sensitive in Python which means 'num' and 'NUM' are two different variables in python.
- Python allows user to assign a single value to several variables simultaneously.
  For example: `a = b = c = 5`
- Here, an integer object is created with the value 5, and all three variables are assigned to the same memory location.
- Multiple variables can be assigned to multiple objects.
  For example: a,b,c = 5,6,"Hello"
- Here, two integer objects with values 5 and 6 are assigned to variables a and b respectively, and one string object with the value "Hello" is assigned to the variable c.

3.4.4 Python Data Types

- There are many data types available in Python. Python has various standard data types to define the operations possible on them and the storage method for each of them.
- Python has 5 standard data types as:
  1. Numbers
  2. String
  3. List
  4. Tuple
  5. Dictionary

1. Python Numbers:

- This data types store numeric values. To number objects, a value is assigned to them.
  For example:

```
var1 = 2
var2 = 15
```

- The reference to a number object can be deleted by using the del statement. The syntax of the del statement is :

```
del var1[,var2[,var3[....,varN]]]
```

- A single object or multiple objects can be deleted by using the del statement.
  For example: del var

```
del var_a, var_b
```

- Python supports 4 different numerical types:
  (i) int (signed integers)
  (ii) long (long integers, can also be represented in octal and hexadecimal)
  (iii) float (floating point real values)
  (iv) complex (complex numbers)
- Python allow to use a lowercase l with long, but it is recommended that only an uppercase L shall be used to avoid confusion with the number 1. Python displays long integers with an uppercase L.

2. Python Strings:

- Python strings are identified as an adjoining set of characters represented in the quotation marks.
- Python allows pairs of single or double quotes.
- The slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from −1 at the end can be used as subset of strings.
- The plus (+) sign is used for the string concatenation operator and the asterisk (*) is used for the repetition operator.

3. Python Lists:

- A list contains items separated by commas which are enclosed within square brackets ([ ]).
- Lists are similar to arrays in C. However, in Python all the items belonging to a list can be different data type.
- The values of the list can be accessed using the slice operator ([ ] and [:]) with indexes.

4. Python Tuples:

- A tuple is another sequence data type which is similar to the list.
- A tuple consists of a number of values separated by commas which are enclosed within parentheses.
- The main difference between list and tuple is Lists are enclosed in brackets ([ ]) and their elements and size can be changed. Tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as **read-only** lists.

5. Python Dictionary :

- Dictionaries in Python are of hash table type. They work like associative arrays found in Perl. It consists of key-value pairs.
- A dictionary keys are usually numbers or strings.
- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([ ]).

3.4.5 Data Type Conversion

- Sometimes, conversion of built-in data types is required. The type name as a function can be used to convert between types.
- In Python, there are various built-in functions available which can perform conversion from one data type to another. Some of them are listed in below table 3.1.

Table 3.1 : Python Built-in Functions for Data Conversion

Sr. No.	Function	Description
1.	**int(x [,base])**	Converts x to an integer. Base specifies the base if x is a string.
2.	**long(x [,base])**	Converts x to a long integer. base specifies the base if x is a string.
3.	**float(x)**	Converts x to a floating-point number.
4.	**complex(real [,imag])**	Creates a complex number.
5.	**str(x)**	Converts object x to a string representation.
6.	**repr(x)**	Converts object x to an expression string.
7.	**eval(str)**	Evaluates a string and returns an object.
8.	**tuple(s)**	Converts s to a tuple.
9.	**list(s)**	Converts s to a list.
10.	**set(s)**	Converts s to a set.
11.	**dict(d)**	Creates a dictionary. d must be a sequence of (key, value) tuples.
12.	**frozenset(s)**	Converts s to a frozen set.
13.	**chr(x)**	Converts an integer to a character.
14.	**unichr(x)**	Converts an integer to a Unicode character.
15.	**ord(x)**	Converts a single character to its integer value.
16.	**hex(x)**	Converts an integer to a hexadecimal string.
17.	**oct(x)**	Converts an integer to an octal string.

- Python supports the following types of operators.

(i) Arithmetic Operators such as:

+ Addition

− Subtraction

* Multiplication

/ Division

% Modulus

** Exponent

//

(ii) Comparison (Relational) Operators such as:

== Equal to

!= No equal to

<> Range

> Greater than

< Less than

>= Greater than and equal to

<= Less than and equal to

(iii) Assignment Operators such as:

=

+ = Add AND

− = Subtract AND

* = Multiply AND

/ = Divide AND

% = Modulus AND

** = Exponent AND

// = Floor Division

(iv) Bitwise Operators such as:

& Binary AND

| Binary OR

^ Binary XOR

~ Binary Ones Complement

<< Binary Left Shift

>> Binary Right Shift

(v) Logical Operators such as:

and Logical AND

or Logical OR

not Logical NOT

(vi) Membership Operators such as: in, not in

(vii) Identity Operators such as: is, is not

3.5 Flow Control Structures

- Flow control structures basically control the flow of program execution depending upon the condition is true or false. The action to be taken is specified in program depending on the outcome is true or false.

- The general format of decision making structure is given below:

Name	Description
Process	Normal Code Statement - 1 input and 1 output. Example would be setting up a variable.
Decision	Condition Block - 1 input and 2 or more potential outputs, makes decision on output based on the condition. Condition example would be, is A > B?
	Direction of the execution flow of the software program. path can be traversed again and again.

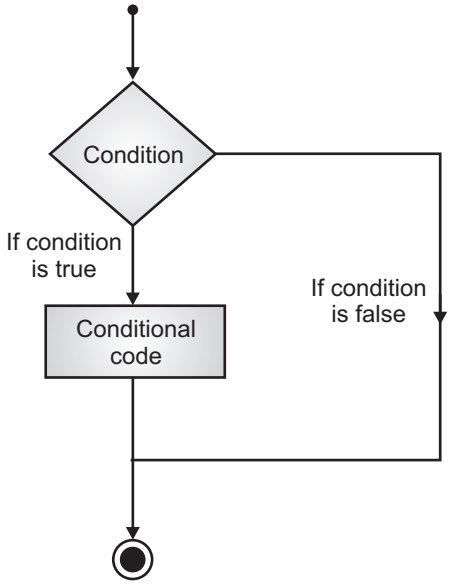


Fig. 3.3: Flow control structure

- As shown in above Fig. 3.3, flow control structures are nothing but decision making structures. It is usually represented in the form of flowchart with specified boxes. There are rectangles, diamonds and arrows. Each box represents a unique action.
- Python assumes any non-zero and non-null values as TRUE condition. If it is either zero or null then it is treated as FALSE value.
- The flow of a Python program is controlled by conditional statements, Loops and function calls.
- Conditional statements includes if...then statements. Loops consists of For loop, While loop etc. There are various Functions available in Python, some of them are discussed here.

(i) If...Then...else

- Sometimes few statements need to be executed only if a particular condition is satisfied. In that case *If.. Then.. else* statement is required.

- In Python, compound statement *if, elif* and *else* is used which will conditionally execute blocks of statements.
- The syntax is as follows:

```
if expression:
   statement(s)
elif expression:
   statement(s)
elif expression:
   statement(s)
...
else:
   statement(s)
```

- The elif and else statements are optional. However, Python does not have a switch statement. For all conditions, *if, elif* and *else* must be used.

```
if y > 0: print "y is positive"
elif y % 2: print "y is negative and even"
else: print "y is odd and negative"
Or it can be written as
if y > 0:
   print "y is positive"
elif y % 2:
   print "y is nagetive and even"
else:
   print "y is odd and positive"
```

(ii) While statement

- The while statement in Python is basically for the repeated execution of a statement or block of statements which are controlled by a conditional expression.
- The syntax for while statement is as below:

```
while expression:
   statement(s)
```

- A while statement also include as *else* condition, *break* and *continuous* statement

```
count = 0
while x > 0:
   x = x // 2   # truncating division
   count += 1
print "The approximate log2 is", count
```

- First loop condition is evaluated. If the condition is false then the *While* statement ends. If loop condition is true then the statement or block of statements which are in loop body are executed.
- When all the statements in the loop finishes executing, the loop condition is evaluated again to check whether another iteration is required or not.
- This process continues until the loop condition is false, then the *While* statement ends.
- The loop body should contain statements which can make the loop condition false, or the loop will never end and it becomes endless loop. If it happens then a *break* statement is raised.
- A loop body ends if it has *return* statement, then whole function ends in this case.

(iii) For statement

- In Python, the *for* statement is used for repeated execution of a statement or block of statements which is controlled by an iterable expression.
- The syntax for *for* statement is as below:

```
for target in iterable:
    statement(s)
```

- Here, keyword **in** is a part of syntax of the *for* statement. The *for* statement can also include an *else* and *break* and *continue* statement.

Iterators:

- An `iterator` is any object `i` such that `i.next()` can be called without any arguments.
- `i.next()` returns the next item of iterator `i`, or, when iterator `i` has no more items, raises a `StopIteration` exception.
- The `for` statement implicitly calls `iter` to get an iterator.
- Python provides built-in functions `range` and `xrange` to generate and return integer sequences. The simplest way to loop *n* times in Python is:

```
for i in xrange(n):
   statement(s)
```

- `range(x)` returns a list whose items are consecutive integers from `0` (included) up to `x` (excluded).
- `range(x,y)` returns a list whose items are consecutive integers from `x` (included) up to `y` (excluded). The result is the empty list if `x` is greater than or equal to `y`.

The break Statement

- The `break` statement can be only used inside a loop body.
- When `break` executes, the loop terminates. If a loop is nested inside other loops, `break` terminates only the innermost nested loop.
- The syntax of loop is as follows:

```
while True: # this loop can never terminate
naturally
```

```
        x = get_next( )
        y = preprocess(x)
    if not keep_looping(x,y): break
    process(x,y)
```

The continue Statement

- The `continue` statement in Python returns the control to the beginning of the while loop.
- The `continue` statement rejects all the remaining statements in the current iteration of the loop and goes back to the top of the loop.
- The `continue` statement can be used in both while and for loops.

The else Clause on Loop Statements

- Both the `while` and `for` statements can use `else` clause.
- The statement after the `else` executes when the loop terminates naturally. However, if the loop terminates prematurely (via `break, return`, or an exception) then no further statements are executed.
- If a loop contains one or more `break` statements, then user has to check whether the loop terminates naturally or prematurely.
- User can use an `else` clause on the loop for this purpose:

```
for y in some_container:
    if is_ok(y): break
       # item y is satisfactory, terminate loop
else:
    print "Warning: no satisfactory item was
    found in container"
y = None
```

The pass Statement

- The body of Python program cannot be empty, it must contain at least one statement. The *pass* statement can be used in that case which performs no action. So when a statement as per syntax is required and nothing is written there then *pass* statement can be used.

```
if condition1(x):
   process1(x)
elif x>33 or condition2(x) and x<5:
   pass      # nothing to be done in this case
elif condition3(x):
   process3(x)
else:
   process_default(x)
```

The try Statement

- In Python, the *try* statement includes *try*, *expect*, *finally* and *else* clauses.
- A *try* statement is used to catch exceptions that might be thrown as program executes.
- The *try* block tries to catch the errors.
- The *try* and except block in Python is used to catch and handle exceptions.

3.6 Functions

- A function is a block of code which only runs when it is called.
- Functions are important as they can process data and can be called as and when required in main program.
- The data can be passed with known parameter into a function. A function can return data as a result.
- A function is a block of organized, reusable code that is used to perform a single action. Functions provide better modularity for the application and a high degree of code reusing.
- Python provides many built-in functions like print(), etc. And users can create their own functions. These functions are called user-defined functions.
- Function block begin with the keyword **def** followed by function name and parentheses (( )). Any input parameter should be put in these parentheses.
- The code block of each function starts with a colon () and is intended.
- The statement return [expression] has to be used at the end of the function.
- The syntax of function is given as below:

```
def functionname(parameters):
    "function_docstring"
    function_suite
    return [expression]
```

- Function consists of **def** which marks the start of the function, function name, parameters (arguments) through which values can be passed to function, a body of function or block of statements a colon (:) to mark end of function and optional return statement to return a value from the function.
- Once the function is defined, it can be called by its name directly from the Python prompt.
- In Python, the parameters (arguments) are passed by reference. That means, if the parameter within a function is changed then the change also reflects back in the calling function.
- A function can be called by using various types of formal arguments as: required arguments, keyword arguments, default arguments, variable-length arguments.
- The *return* statement exits a function.

3.6.1 Scope of Variables

- In Python, all variables in a program may not be accessible at all locations of that program. It depends on the variable declaration.

- The scope of variable determines accessibility of that variable. The variable can be declared as Global variables or Local variables.
- Variables which are defined inside the function have local scope and those which are defined outside the function have global scope.
- Local variables can be accessed only inside the function in which they are declared. Global variables can be accessed at any location of a program by any function.

3.6.2 Types of Functions

- There are two types of functions: Built-in functions and User-defined functions.
- Built-in functions are built in Python. User-defined functions are defined by the users.
- Built-in functions of Python are listed in below table.

Table 3.2 : Python Built-in Functions

Built-in Functions				
abs ( )	delattr ( )	hash ( )	memoryview ( )	set ( )
all ( )	dict ( )	help ( )	min ( )	setattr ( )
any ( )	dir ( )	hex ( )	next ( )	slice ( )
ascii ( )	divmod ( )	id ( )	object ( )	sorted ( )
bin ( )	enumerate ( )	input ( )	oct ( )	staticmethod ( )
bool ( )	eval ( )	int ( )	open ( )	str ( )
breakpoint ( )	exec ( )	isinstance ( )	ord ( )	sum ( )
bytearray ( )	filter ( )	issubclass ( )	pow ( )	super ( )
bytes ( )	float ( )	iter ( )	print ( )	tuple ( )
callable ( )	format ( )	len ( )	property ( )	type ( )
chr ( )	frozenset ( )	list ( )	range ( )	vars ( )
classmethod ( )	getattr ( )	locals ( )	repr ( )	zip ( )
compile ( )	globals ( )	map ( )	reversed ( )	_import_ ( )
complex ( )	hasattr ( )	max ( )	round ( )	

[I] I/O Function (GPIO, Digital):

- Python program has to obtain data from user. It can be taken as an input from keyboard. In Python, the input function is available as *input ( )* which can be written on Python prompt. .

  For example :

```
>>> s = input()
  Hello
  >>> s
 'Hello'
```

- The output can be written on console by using *print ( )* function as:

```
print(<obj>, ..., <obj>)
```

For example :

```
>>> print("Hello")
Hello
```

GPIO:

- General Purpose Input Output (GPIO) pins of Raspberry Pi can be used as input or output. Through GPIO pins, Raspberry Pi connects to outside world and can access input and output.
- GPIO pins of Raspberry Pi are the physical interface between Pi and the outside world.
- GPIO pins can be programmed according to user's need to interact with external devices. For example: to read the state of a physical switch, any GPIO pin can be configured as input pin and read the switch status to make decision.
- To use Raspberry Pi GPIO pins in Python, **RPi.GPIO** package has to be imported. This **RPi.GPIO** is already installed on Raspbian OS. So, it does not need to install externally. The library has to be included in the Python program as:

  import RPi.GPIO as GPIO
- The GPIO pin number and physical pin number of Raspberry Pi are different. In GPIO numbering, pin number refers to number on Broadcom SoC and in physical numbering, pin number refers to the pin 40-pin header on Raspberry Pi board.
- So, the GPIO pins of Raspberry Pi has to be defined. They can be defined as:

  GPIO.setmode (Pin Numbering System)
- In RiPi.GPIO, GPIO numbering is identified by **BCM** whereas Physical numbering is identified by **BOARD.**
- Pin Numbering System = BOARD/BCM
- E.g. If user use pin number 40 of P1 header as a GPIO pin then it has to be configured as output.

In BCM,

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(21, GPIO.OUT)
```

In BOARD,

```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(40, GPIO.OUT)
```

GPIO.setup (channel, direction, initial value, pull up/pull down)

This function is used to set the direction of GPIO pin as an input/output.

channel: GPIO pin number as per numbering system.

direction: set direction of GPIO pin as either Input or Output.

initial value: can provide initial value.

pull up/pull down: enable pull up or pull down if required.

Few examples are given as follows:

- GPIO as Output
  GPIO.setup(channel, GPIO.OUT)
- GPIO as Input
  GPIO.setup(channel, GPIO.IN)
- GPIO as Output with initial value
  GPIO.setup(channel, GPIO.OUT, initial=GPIO.HIGH)
- GPIO as Input with Pull up resistor
  GPIO.setup(channel, GPIO.IN, pull_up_down = GPIO.PUD_UP)

GPIO.output(channel, state)

This function is used to set the output state of GPIO pin.

channel: GPIO pin number as per numbering system.

state: Output state i.e. HIGH or LOW of GPIO pin.

e.g.

GPIO.output(7, GPIO.HIGH)

GPIO.input(channel)

This function is used to read the value of GPIO pin.

e.g.

GPIO.input(9)

Example:

Turn ON and OFF LED using Python on Raspberry Pi. Switch is used to control the LED ON-OFF.

Python Program :

```
import RPi.GPIO as GPIO     #import RPi.GPIO module

LED = 32                    #pin no. as per BOARD, GPIO18 as per BCM
Switch_input = 29           #pin no. as per BOARD, GPIO27 as per BCM
GPIO.setwarnings(False)     #disable warnings
GPIO.setmode(GPIO.BOARD)    #set pin numbering format
GPIO.setup(LED, GPIO.OUT)   #set GPIO as output
GPIO.setup(Switch_input,GPIO.IN,pull_up_down=GPIO.PUD_UP)

while True:
    if(GPIO.input(Switch_input)):
        GPIO.output(LED,GPIO.LOW)
    else:
        GPIO.output(LED,GPIO.HIGH)
```

[II] Time Functions:

- Date and time are very important in everyday life. So all the programming languages have date and time functions.
- Python provides three modules: datetime, time and calendar which helps user to deal with dates, time, duration and calendars.
- The 'datetime' module consists of four important classes as 'datetime', 'date', 'time', and 'timedelta'.
- The 'datetime' class handles a combination of date and time which has attributes as year, month, day, hour, minute, second, microsecond and tzinfo.
- The data class handles dates of Gregorian calendar and has year, month and day attributes.
- The time class handles time consuming per day that means 24 × 60 × 60 seconds. It has hour, minute, second, microsecond and tzinfo attributes.
- The timedelta class handles the duration. Duration can be difference between two; date, time or datetime instances.

The epoch

- The '*epoch*' is the point where the time starts. This point considers the 0.0 hours of January 1st of the current year.
- **For example:** Python program to measure the time in seconds from epoch is as follows:

```
#knowing the time since the epoch
import time
epoch = time.time()    # call time() function of time module
print epoch()
```

The output of the program is:

```
C:\python ep.py
1462077746.917558
```

Above time is in seconds from epoch.

(i) The *time* function: Python consists of a function named *'time'* to handle time related tasks. For this, first the time module has to be imported by using –

```
import time
```

The time ( ) function returns the number of seconds passed since epoch.

(ii) The time.ctime (): The time.ctime () function considers seconds passed since epoch as an argument and returns string representing local time which is more understandable to user.

```
import time
# seconds passed since epoch
seconds = 1545925769.9618232
local_time = time.ctime(seconds)
print("Local time:", local_time)
```

The output of the program is as :

```
Local time: Thu Dec 27 15:49:29 2018
```

(iii) The time.clock (): The time.clock () function return the processor time which is used to test the performance testing/benchmarking as:

```
time.clock()
```

The `clock()` function returns the right time taken by the program.

(iv) The time.sleep (): The time.sleep() function pause the current execution for specified number of seconds. It passes a floating point value as input to get more precise sleep time.

The sleep () function can be used in programs where user need to wait for a file to finish closing.

For example :

```
import time
# using ctime() to display present time
print ("Time starts from : ",end="")
print (time.ctime())
# using sleep() to suspend execution
print ('Waiting for 5 sec.')
time.sleep(5)
# using ctime() to show present time
print ("Time ends at : ",end="")
print (time.ctime())
```

The output of above program is :

```
Time starts from : Fri Mar 22 20:00:00 2019
Waiting for 5 sec.
Time ends at : Fri Mar 22 20:00:05 2019
```

(v) The time.struct_time class: The time.struct_time is used for data structure present in the time module. It is named under tuple interface and can be accessible via index or the attribute name. Its syntax is as:

```
time.struct_time
```

This class is used when the specific field of a date has to be accessed.

This class provides functions such as localtime (), gmtime () and return struct_time objects.

For example :

```
import time
print('Current local time:', time.ctime())
t = time.localtime()
print('Day of month:', t.tm_mday)
```

```
print('Day of week :', t.tm_wday)
print('Day of year :', t.tm_yday)
```

The output of the program is

```
Current local time: Fri Mar 22 20:10:25 2019
Day of month: 22
Day of week : 4
Day of year : 81
```

(vi) The time.strftime (): The time.strftime () function takes a tuple or struct_time in the second argument and converts to a string as per the format specified in the first argument. Its syntax is as:

```
time.strftime()
```

For example:

```
import time
now = time.localtime(time.time())
print("Current date time is: ",time.asctime(now))
print(time.strftime("%y/%m/%d %H:%M", now))
print(time.strftime("%a %b %d", now))
print(time.strftime("%c", now))
print(time.strftime("%I %p", now))
print(time.strftime("%Y-%m-%d %H:%M:%S %Z", now))
```

The output of the program is

```
Current date time is: Fri Mar 22 20:13:43 2019
19/03/22 20:13
Fri Mar 22
Fri Mar 22 20:13:43 2019
08 PM
2019-03-22 20:13:43 India Standard Time
```

- Python has various format codes used in strftime as given in table 3.2.

Table 3.2 : Python Format Codes

Format Code	Meaning	Example
%a	Weekday as an abbreviated name.	Sun, Mon, …., Sat
%A	Weekday as full name.	Sunday, Monday, …., Saturday
%w	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.	0, 1, …., 6

… (Contd.)

Format Code	Meaning	Example
%d	Day of the month as a zero-padded decimal number.	01, 02, …., 31
%b	Month as an abbreviated name.	Jan, Feb., …., Dec.
%B	Month as full name.	January, February, …., December
%m	Month as zero-padded decimal number.	01, 02, …., 12.
%y	Year without century as a zero-padded decimal number.	00, 01, …., 99
%Y	Year with century as decimal number.	0001, 0002, …., 2016, …., 9999
%H	Hour (24-hour clock) as zero-padded decimal number.	00, 01, ….,23
%I	Hour (12-hour clock) as a zero-padded decimal number.	01, 02, …., 12
%p	Either AM or PM.	AM, PM
%M	Minute as a zero-padded decimal number.	00, 01, …., 59
%S	Second as a zero-padded decimal number.	00, 01, …., 59
%f	Microsecond as a decimal number, zero-padded on the left.	000000, 000001, …., 999999
%Z	Time zone name.	(empty), UTC, EST, CST
%j	Day of the year as a zero-padded decimal number.	001, 002, …., 366
%U	Week number of the year (Sunday as the first day of the week) as a zero-padded decimal number. All days in a new year preceding the first Sunday are considered to be in week 0.	00, 01, …., 53
%W	Week number of the year (Monday as the first day of the week) as a decimal number. All days in a new year preceding the first Monday are considered to be in week 0.	00, 01, …., 53
%c	Appropriate date and time representation.	Tue Aug 16 21:30:00 1988
%x	Appropriate date representation.	08/16/88 (None); 08/16/1988 (en_US)
%X	Appropriate time representation.	21:30:00 (en_US)
%%	A single % character.	%

(vii) The timedelta: The timedelta class is useful to find duration such as differences between two dates or finding the date after adding a period to current date.

It is also used to find the future date or previous dates using timedelta class.

The format of timedelta is:

```
datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0,
minutes=0, hours=0, weeks=0)
```

Days, seconds and microseconds are only stored internally in the *timedelta* object. Other arguments are converted to those units as:

- A millisecond is converted to 100 microseconds.
- A minute is converted to 60 seconds.
- An hour is converted to 3600 seconds.
- A week is converted to 7 days.

Calendar Module

The *'calendar'* module is used to create calendar of any month or year. It is also used to know whether the year is a leap year or not by using *isleap ()* function.

For example :

```
# to test whether leap year or not
from calendar import *
y = int (input ('entre year:  '))
if (isleap(y)):
   print (y,   '   is leap year')
else:
   print (y, '   is not leap year')
```

The output of the program is :

```
C:\>python leap.py
Entre year: 2020
2020 is leap year
```

The perf_counter() and process_time()

- The perf_counter() and process_time() functions of 'time' module are used to measure the time difference between two points in a program. Thus, these two functions are used to measure the time taken for execution of a python program.
- perf_counter () function returns the time duration in fractional seconds. It measures the time taken by the program to execute a group of statements. It includes the time elapsed during sleep of the processor.
- process_time () function also return the time duration in fractional seconds. It measures the total time taken by the program and CPU in executing a group of statements. It will not include the time elapsed during the sleep of the processor.

The syntax are as:

```
t1 = perf_counter ()
t2 = process_time ()
```

[III] Library Functions:

- The standard library of Python is very extensive. It offers a wide range of functions.
- The library contains built-in modules which are written in 'C'. These built-in functions give access to system functionality such as I/O file.
- Python library also provides various modules which can give solutions to many problems in programming.
- The Python installers for Windows platform include the entire standard library and also many additional components. For Unix operating system, Python provides collection of packages.
- Also several thousand components from individual programs and modules are available along with the standard library.
- Some of the built-in functions are tabulated in Table 3.3.
- The Python standard library is available on their official website docs.python.org>library.
- Libraries are maintained by group of contributors and it is made available to all users. This type of collaboratively maintained software is known as open source. Python is an open source.
- In Python, library has to be imported to access library functions. To import library, the code has to be downloaded into the computer where Python is running and then import it.
- Python is an ocean of libraries which are used for various purposes. Top 10 Python libraries which are widely used are: TensorFlow, Scikit-Learn, Numpy, Keras, PyTorch, LightGBM, Eli5, SciPy, Theano, Pandas.

Commonly used Python built-in functions are listed below:

Table 3.3 : Commonly used Python Built-in Functions

Function	Description
Python abs()	Returns absolute value of a number.
Python any()	Checks if any Element of an Iterable is True.
Python all()	Returns true when all elements in iterable are true.
Python ascii()	Returns string containing printable representation.
Python bin()	Converts integer to binary string.
Python bool()	Converts a Value to Boolean.
Python bytearray()	Returns array of given byte size.
Python callable()	Checks if the object is callable.

... (Contd.)

Function	Description
Python bytes()	Returns immutable bytes object.
Python chr()	Returns a character (a string) from an Integer.
Python compile()	Returns a python code object.
Python classmethod()	Returns class method for given function.
Python complex()	Creates a complex number.
Python delattr()	Deletes attribute from the object.
Python dict()	Creates a Dictionary.
Python dir()	Tries to return attributes of object.
Python divmod()	Returns a tuple of quotient and remainder.
Python enumerate()	Returns an enumerate object.
Python staticmethod()	Transforms a method into a static method.
Python filter()	Constructs iterator from elements which are true.
Python eval()	Runs python code within program.
Python float()	Returns floating point number from number, string.
Python format()	Returns formatted representation of a value.
Python frozenset()	Returns immutable frozenset object.
Python getattr()	Returns value of named attribute of an object.
Python globals()	Returns dictionary of current global symbol table.
Python exec()	Executes Dynamically Created Program
Python hasattr()	Returns whether object has named attribute.
Python help()	Invokes the built-in help system.
Python hex()	Converts to Integer to Hexadecimal.
Python hash()	Returns hash value of an object.
Python input()	Reads and returns a line of string.
Python id()	Returns Identity of an object.
Python isinstance()	Checks if an object is an Instance of Class.
Python int()	Returns integer from a number or string.
Python issubclass()	Checks if a Class is Subclass of another Class.
Python iter()	Returns an iterator.
Python list()	Creates a list in Python.

... (Contd.)

Function	Description
Python locals()	Returns dictionary of a current local symbol table.
Python len()	Returns length of an object.
Python max()	Returns the largest item.
Python min()	Returns the smallest value.
Python map()	Applies function and returns a list.
Python next()	Retrieves next item from the iterator.
Python memoryview()	Returns memory view of an argument.
Python object()	Creates a featureless object.
Python oct()	Returns the octal representation of an integer.
Python ord()	Returns an integer of the Unicode character.
Python open()	Returns a file object.
Python pow()	Returns the power of a number.
Python print()	Prints the given object.
Python property()	Returns the property attribute.
Python range()	Returns sequence of integers between start and stop.
Python repr()	Returns a printable representation of the object.
Python reversed()	Returns the reversed iterator of a sequence.
Python round()	Rounds a number to specified decimals.
Python set()	Constructs and returns a set.
Python setattr()	Sets the value of an attribute of an object.
Python slice()	Returns a slice object.
Python sorted()	Returns a sorted list from the given iterable.
Python str()	Returns the string version of the object.
Python sum()	Adds items of an iterable.
Python tuple()	Returns a tuple.
Python type()	Returns the type of the object.
Python vars()	Returns the __dict__ attribute.
Python zip()	Returns an iterator of tuples.
Python __import__()	Function called by the import statement.
Python super()	Returns a proxy object of the base class.

... (Contd.)

Function	Description
Python Dictionary clear()	Removes all Items.
Python Dictionary copy()	Returns shallow copy of a dictionary.
Python Dictionary fromkeys()	Creates dictionary from given sequence.
Python Dictionary get()	Returns value of the key.
Python Dictionary items()	Returns view of dictionary's (key, value) pair.
Python Dictionary keys()	Returns view object of all keys.
Python Dictionary popitem()	Returns and removes latest element from dictionary.
Python Dictionary setdefault()	Inserts key with a value if key is not present.
Python Dictionary pop()	Removes and returns element having given key.
Python Dictionary values()	Returns view of all values in dictionary.
Python Dictionary update()	Updates the dictionary.
Python List append()	Adds a single element to the end of the list.
Python List extend()	Adds iterable elements to the end of the list.
Python List insert()	Insert an element to the list.
Python List remove()	Removes item from the list.
Python List index()	Returns the index of the element in the list.
Python List count()	Returns count of the element in the list.
Python List pop()	Removes element at the given index.
Python List reverse()	Reverses the list.
Python List sort()	Sorts elements of a list.
Python List copy()	Returns a shallow copy of the list.
Python List clear()	Removes all items from the list.

3.7 Basic Arithmetic Programs: Addition, Subtraction, Multiplication, Division

- As mentioned earlier in this chapter, Python is a cross-platform programming language that means it can run on multiple platforms like Windows, MacOs, Linux etc.
- Python program can be typed and run on command prompt or in IDEL.
- On command prompt, it immediately returns the results of command that user has entered. In IDEL, the program can be saved and then run to see the outputs.
- There are different ways in which Python program can be run as:
  1. Interactive Mode
  2. Command Line

3. Text Editor (VS Code)
4. IDE (PyCharm)

1. Interactive Mode:

- In Interactive mode, the program can be run line by line in a sequence. To entre in an interactive mode, you will have to open Command Prompt on Windows and type 'python' and press enter.

2. Command line:

- To run Python script stored in a *' .py'* file in command line, user has to write *'python'* keyword before the file name in the command line.

3. Text Editor (VS Code):

- To run Python script on a text editor such as VS Code (Visual Studio Code), following steps have to followed:
- Go in the extension section or press *' ctrl+shift+X'* on Windows, then search and install extension '*Python*' and '*Code Runner*'. Restart *VS Code* after that. Then create a new file with the name '*helllo.py*' and write a code.

  For example: print ('Hello')
- Then right click on the text area and select the option '*Run Code*' or press '*ctrl+Alt+N*' to run the program.

4. IDE (PyCharm):

- Python script can be run on a IDE (Integrated Development Environment) such as PyCharm, following steps have to be followed for this:
  - Create a new project.
  - Give a name to that project as 'GfG' and click on Create.
  - Select the root directory with the project name as specified in the last step. **Right click** on it, go in **New** and click on '**Python file**' option.
  - Then give the name of the file as '**hello**' (any name can be specified).
  - This will create a '**hello.py**' file in the project root directory.
  - Here, the extension .py need not be specified, as it will take it automatically.

3.7.1 Python Programs for Addition, Subtraction, Division and Multiplication

[I] Python program for addition of 2 numbers:

```
# Python program for addition of two numbers

num1 = 1.5
num2 = 3.5

# Add two numbers
sum = num1 + num2
# Display the sum
print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
```

Output

```
The sum of 1.5 and 3.5 is 5.0
```

[II] Python program for subtraction of 2 numbers:

```
# Python program for subtraction of two numbers

num1 = 5.0
num2 = 3.0
# subtract two numbers
sub = num1 - num2

# Display the sub
print('The subtraction of {0} and {1} is {2}'.format(num1, num2, sub))
```

Output

```
The subtraction of 5.0 and 3.0 is 2.0
```

[III] Python program for multiplication of 2 numbers:

```
# Python program for multiplication of two numbers

num1 = 5.0
num2 = 3.0
# multiply two numbers
mul = num1 * num2

# Display the mul
print('The multiplication of {0} and {1} is {2}'.format(num1, num2,
mul))
```

Output

```
The multiplication of 5.0 and 3.0 is 15.0
```

[IV] Python program for division of 2 numbers:

```
# Python program for division of two numbers

num1 = 15.0
num2 = 3.0
# division of two numbers
div = num1/num2

# Display the div
print('The division of {0} and {1} is {2}'.format(num1, num2, div))
```

Output

```
The division of 15.0 and 3.0 is 5.0
```

[V] Python program for addition, subtraction, multiplication and division using functions :

```
#Addition function
def add(x,y):
    s = x + y
    print("Addition is: ",s)

#Subtraction function
def subtract(x,y):
    s = x - y
    print("Subtraction is: ",s)

#Multiplication function
def multiply(x,y):
    s = x*y
    print("Multiplication is: ",s)

#Division function
def division(x,y):
    s = x/y
    print("Division is: ",s)

#Ask for user input
x = int(input("Enter x: "))
y = int(input("Enter y: "))

#Function call
add(x,y)
subtract(x,y)
multiply(x,y)
divide(x,y)
```

Output:

```
Enter x: 20
Enter y: 5
Addition is:  25
Subtraction is:  15
Multiplication is:  100
Division is:  4.0
```

Think Over It

- What is the difference between Raspberry Pi and Tinker board?
- How can real data be accessed through Raspberry Pi for FFT?
- How to generate PWM using Raspberry Pi?
- What is the difference between Class method and Static method ?
- What is GIL? How to use it in Python?
- What are Decorators in Python?
- What are Pickling and Unpickling in Python?

Points to Remember

- Raspberry Pi OS which is formerly known as Raspbian is a Debian-based operating system for Raspberry Pi.
- Raspbian is the most popular operating system for Raspberry Pi. The installation process of Raspbian is same for all models of Raspberry Pi.
- In Raspbian, thousands of pre built libraries to perform many tasks and optimize the OS are available.
- Raspbian is the official Raspberry Pi OS, however other OS are also available which can run on Raspberry Pi.
- Python is a general purpose interpreted, interactive, object-oriented and high-level programming language.
- It was created by Guido van Rossum during 1985-1990 at the National Research Institute for Mathematics and Computer Science in the Netherlands.
- Python is copyrighted. Python source code is also available under the GNU General Public License (GPL).
- Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation and it has fewer syntactical constructions than other languages.
- The documentation of Python is available on heet://www.python.org/doc/. The documentation is available in HTML, PDF and PostScript formats.
- Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when a value to a variable is assigned.
- Python has 5 standard data types as:
  1. Numbers
  2. String

3. List
4. Tuple
5. Dictionary

- The main difference between list and tuple is Lists are enclosed in brackets ([ ]) and their elements and size can be changed. Tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read-only lists.
- The flow of a Python program is controlled by conditional statements, loops and function calls.
- Conditional statements includes if...then statements. Loops consists of For loop, While loop etc.
- The body of Python program cannot be empty, it must contain at least one statement. The pass statement can be used in that case which performs no action.
- A try statement is used to catch exceptions that might be thrown as program executes.
- The try block tries to catch the errors.
- A function is a block of code which only runs when it is called.
- Functions are important as they can process data and can be called as and when required in main program.
- Function block begin with the keyword **def** followed by function name and parentheses (()). Any input parameter should be put in these parentheses.
- The *return* statement exit from the function.
- The scope of variable determines accessibility of that variable. The variable can be declared as Global variables or Local variables.
- Local variables can be accessed only inside the function in which they are declared. Global variables can be accessed at any location of a program by any function.
- Python provides three modules: datetime, time and calendar which helps user to deal with dates, time, duration and calendars.
- The 'epoch' is the point where the time starts.
- Python consists of a function named *'time'* to handle time related tasks. For this, first the time module has to be imported by using – `import time`.
- The *'calendar'* module is used to create calendar of any month or year.
- The perf_counter() and process_time() functions of 'time' module are used to measure the time difference between two points in a program.
- The standard library of Python is very extensive. It offers a wide range of functions.

Exercises

[A] True or False :

1. Raspberry Pi OS which is formerly known as Raspbian is a Debian-based operating system for Raspberry Pi.
2. Raspbian is not the official Raspberry Pi OS.
3. Python is completely object oriented.
4. Python variables require explicit declaration to reserve memory space.
5. Python allows pairs of single or double quotes.
6. The main difference between list and tuple is Lists are enclosed in brackets ([ ]) and Tuples are enclosed in parentheses (( )).

[B] Multiple Choice Questions :

1. …… are the types of Operating systems used for Raspberry Pi.
   (a) Raspbian (b) DietPi
   (c) LIBREELC (d) All of above
2. …… is the official Operating system for Raspberry Pi.
   (a) Raspbian (b) DietPi
   (c) LIBREELC (d) None of these
3. Python is …… high level language.
   (a) interpreted (b) object oriented
   (c) interactive (d) All of above
4. Python can be installed on ……
   (a) windows (b) Linux
   (c) maOS (d) All of above
5. Python has …… data types.
   (a) 3 (b) 4
   (c) 5 (d) 6

[C] Fill in the Blanks:

1. Raspberry Pi OS is formerly known as _____
2. Python is a general purpose _____, _____, object-oriented and high-level programming language.

3. The _____ block tries to catch the errors.
4. Function block begin with the keyword _____ followed by function name and parentheses (()).
5. Python provides three modules: _____, _____ and _____which helps user to deal with dates, time, duration and calendars.
6. The '_____' is the point where the time starts in 'time' module used in Python program.

[D] Short Answer Questions:

1. Which OS is recommended for Raspberry Pi?
2. What is Raspbian?
3. List atleast 4 different OS used for Raspberry Pi.
4. List atleast 4 features of Raspbain OS.
5. What is Python?
6. List atleast 4 features of Python.
7. What are the different ways to install Python?
8. What are Python variables?
9. List data types used in Python.
10. What are Python Tuples?
11. What is the difference between Lists and Tuples?
12. What is 'function' and how function is defined in Pyhton?
13. What are the flow control structures used in Python?
14. What is 'break' and 'pass' statements used in Python?
15. What is the use of 'time' function?

[E] Long Answer Questions:

1. What is Raspbian? List features of Raspbian.
2. Explain installation steps of Raspbian.
3. List various types of operating systems used for Raspberry Pi. Give features of at two OS.
4. What is Python? List features of Python.
5. Describe various types of Python installation.
6. Write a short note on Python variables.
7. Write a short note of various Data types used in Python.

8. Give at least 5 examples of data type conversion functions.
9. Describe flow control structures used in Python with suitable example.
10. Explain if else, for and while loops used in Python.
11. Explain following:
    (a) Break statement
    (b) Pass statement
    (c) Continue statement
    (d) Try statement
    (e) Range
12. What is the use of functions? Give at least 3 examples of functions used in Python.
13. Write a short note on variables used in Python.
14. Explain GPIO functions.
15. Write a short note on GPIO functions.
16. Explain various ways and steps of Python installation.
17. What is 'time' function? Explain various modules used in 'time' function.
18. Explain time.ctime(), time.clock(), time.sleep(), time.struct_time, time.strftime (), modules of 'time' function.
19. What is timedelta class ? Explain calendar function used in Python.
20. Write a short not on Library function used in Python.
21. Write a Python program for the addition of two numbers. Write output of the program.
22. Write a Python program for the subtraction of two numbers. Write output of the program.
23. Write a Python program for the multiplication of two numbers. Write output of the program.
24. Write a Python program for the division of two numbers. Write output of the program.

Answers

[A] True or False :

(1)	True	(2)	False	(3)	True
(4)	False	(5)	True	(6)	True

[B] Multiple Choice Questions :

1. (d)	2. (a)	3. (d)	4. (d)	5. (c)

[C] Fill in the blanks :

1. Raspbain
2. interpreted, interactive
3. try
4. def
5. Datetime, time and calendar
6. epoch

Unit 4...

Interfacing of Devices Using Python Programming

Introduction

- In previous chapters, the details of Raspberry Pi board, its pin configuration, function of each pin were discussed. Also the introduction of Python scripting language, functions used in Python and Python programs for addition, subtraction, multiplication and division were discussed.
- Raspberry Pi has General Purpose Input/output (GPIO) pins which can be used to interface many input output devices. The devices like LEDs, Sensors, display, camera etc. can be interfaced through these GPIO pins.
- Raspberry Pi can be programmed using Python scripting language and various I/O devices can be controlled by programming Raspberry Pi.
- In this chapter, interfacing of LED, switch, LCD, camera, serial communication GSM, ultrasonic sensor, PIR and finger print reader are discussed. The Python programs for all these I/O devices are also given in this chapter.
- Raspberry Pi has 40 pins GPIO connector. The latest version of the Raspberry Pi series, Raspberry Pi Model B, has 40 GPIO pins which include power pins and ground pins.
- Fig. 4.1 shows Raspberry Pi Model B board.
- As discussed in chapter 3, the GPIO pin number and physical pin number of Raspberry Pi are different.
- In GPIO numbering, pin number refers to number on Broadcom SoC and in physical numbering, pin number refers to the pin 40-pin header on Raspberry Pi board. So, the GPIO pins of Raspberry Pi have to be defined.
- As shown in Fig. 4.1, the numbers in the centre which are circled are the physical pins of Raspberry Pi. These pins are known as Board Pins or numbers. The GPIO numbers such as Physical Pin 3 is GPIO 2 which is seen by the processor. These numbering is called as GPIO numbering or BCM numbering. So, while programming Raspberry Pi, user has to take care about these numbering and decide which type of numbering is used in programming.

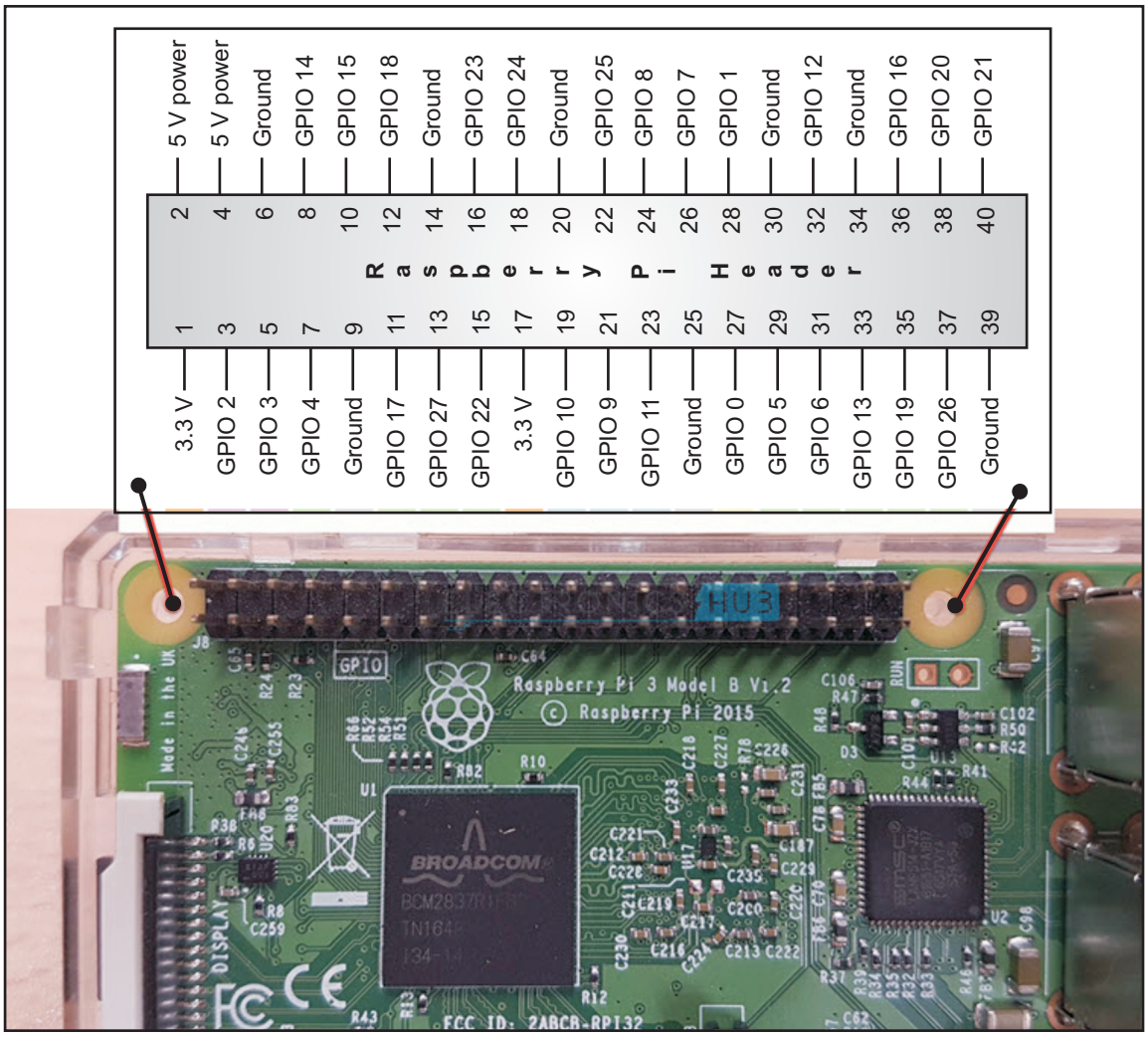


Fig. 4.1: Raspberry Pi Model B

4.1 Python Programming For Raspberry Pi

- Python is an interpreted language that means a Python program or script can be directly executed rather than compiling it into machine code.
- Interpreted languages such as Python are easy and quicker to program with more benefits such as being an interpreter, it identify the type of variable whether it is a number or list or string. Interpreter itself can identify the type of variable.
- Python interpreter can be run in two ways: as an interactive shell to execute individual commands or as a command-line program to execute standalone programs or scripts.
- The integrated development environment (IDE) bundled with Python and the Raspberry Pi is called as IDEL.
- The output of IDEL is very slow as compared with output of command-line program.

- The installation of Python was discussed in previous chapter. Some of the popular modules useful on the Raspberry Pi are shown in below table 4.1:

Table 4.1: Few Popular Packages Used Particularly for Raspberry Pi

Module	Description	URL	Package name
RPi, GPIO	Access to GPIO pins	sourceforge.net/ projects/raspberry-gpio-python	python-rpi.gpio
GPIOzero	Simplified access to GPIO pins	gpiozero.readthedocs.org	python-gpio-zero
Pygame	Gaming framework	pygame.org	python-pygame
SimpleCV	Easy API for Computer Vision	simplecv.org	No package
SciPy	Scientific computing	www.scipy.org	python-scipy
NumPy	The numerical underpinings of Scipy	numpy.scipy.org	python-numpy
Flask	Microframework for web development	flask.pocoo.org	python-flask
Feedparser	Atom and RSS feed parser	pypi.python.org/pypi/ feedparser	No package
Requests	"HTTP for Humans"	docs.python-requests.org	python-requests
PIL	Image processing	www.pythonware.com/ products/pil/	python-imaging
wxPython	GUI framework	wxpython.org	python-wxgtk2.8
pySerial	Access to serial port	https://github.com/pyserial/ pyserial	python-serial
PyUSB	FTDI-USB interface	bleyer.org/pyusb	No package

- To use one of the packages mentioned in table, first that module has to be downloaded, configured and installed. For example: numPy module can be installed as follows:

  sudo apt-get install python-numpy

4.1.1 Setting Raspberry Pi Module

- The Raspberry Pi is actually minicomputer which is capable of doing many things. The pin configuration of Raspberry Pi was discussed in chapter 3. Raspberry Pi can be programmed using Python.
- To write a Python program for Raspberry, first Operating System has to be installed and then Raspberry has to be configured.

- Install Raspbian with Desktop and connect keyboard, mouse and monitor to Raspberry Pi. Windows, macOS or Linux can be used for installing OS.
- If Raspbian Lite is installed then keyboard, mouse and monitor are not required. This will allow getting a terminal into Raspberry Pi on another computer.

4.1.2 Steps for Booting Raspberry Pi

- Booting Raspberry Pi for the first time include following steps:
  - Push the microSD card into the socket on the bottom of the board.
  - Plug in a USB keyboard and mouse.
  - Plug the HDMI output into TV or monitor. Make sure that, monitor is ON and set to the correct input.
  - Plug in power supply and switch ON the board.
- Once the board is ON, Raspberry Pi logo appears on screen. While configuring the Raspberry Pi board first time, Raspbian desktop environment appears on screen.
- Few things have to be set with Raspbian Pi Configuration Tool. To open it, click → Menu Preferences → Raspberry Pi Configuration as shown in below Fig. 4.2.

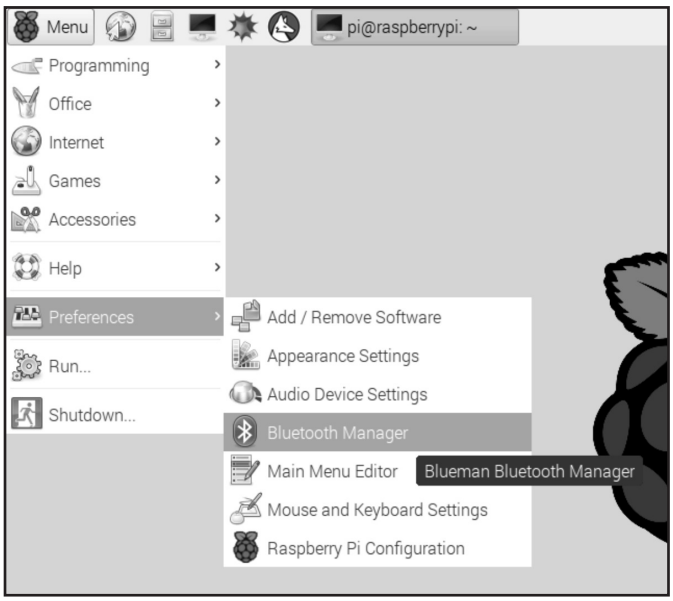


Fig. 4.2: Raspberry Pi configuration tool

- The Raspberry Pi configuration tool allows user to change many important settings on Raspberry Pi such as:

  System → Expand Filesystem, change password, Boot, Overscan etc.

  Interface → SSH, GPU memory, Overclock

  Localization → Set keyboard, Set Locale, Set Timezone etc.

- A typical laptop or desktop will have additional hardware to set the date and time; however Raspberry Pi does not have such hardware. Raspbian is configured automatically and get synchronized its time and date with a network Time Protocol (NTP) server when plugged into a network.
- To write a Python program for Raspberry Pi, GPIO library is required. **In Raspbian, Python and GPIO library are preinstalled.**

4.1.3 First Python Program on Raspberry Pi

- To write first Python program, open Python's IDEL editor, In Menu →Programming → and click on Python 3 as shown in below Fig. 4.3.

Fig. 4.3: Python's IDEL editor

- The message containing Python version, date, time, license will appear on screen.
- Type "print ("Hello !")" into the Python window and press enter, 'Hello !' word will appear on blue color on screen.
- For every new program, open Python's IDLE editor and open a new window (CTRL + N), enter the code below and save it. To run the Python program, type `sudo python filename.py`

4.2 Interfacing of LED to Raspberry Pi

- Raspberry Pi has 17 General Purpose Input Output pins through which any I/O devices can be connected.
- Pin number 3, 5, 7, 8, 10, 11, 12, 13, 15, 16, 18, 19, 21, 22, 23, 24, 26 are the GPIO pins of Raspberry Pi.
- LED can be interfaced or connected to any of these I/O pins.
- Connect the LED between ground and 25 (GPIO) with a 1 kΩ resistor in series as shown in below Fig. 4.4.

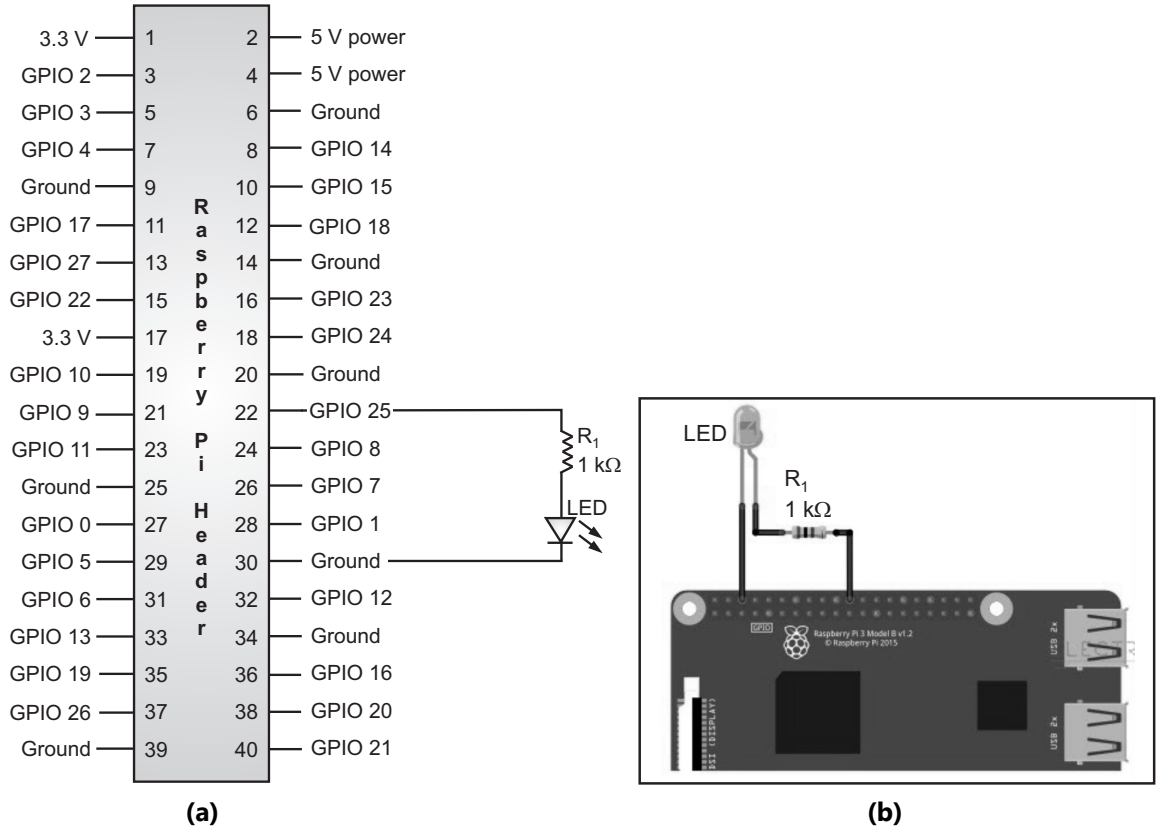


Fig. 4.4 : Interfacing LED to Raspberry Pi

- As shown in above Fig. 4.4, the anode of the LED is connected to GPIO25 (physical pin 22) with a series resistance of 1 kΩ value. The cathode of the LED is connected to GND pin. The GPIO25 pin has to be configured as an output pin so that Raspberry Pi can drive the LED.
- If GPIO25 pin made High for some time and LOW for some time and if this process is kept in a loop then LED will start blinking.
- A Python program is written to make GPIO25 pin high and low.

4.2.1 Algorithm for Blinking LED

- Import the code needed for GPIO control.
- Import the code needed for the sleep function.
- Use the chip's signal numbers.
- Set pin 25 as an output pin.
- Create an infinite loop consisting the required code.
- Turn the LED ON.
- Wait for one second.
- Turn the LED OFF.
- Wait for one second.

4.2.2 Procedure of Interfacing and Blinking of LED

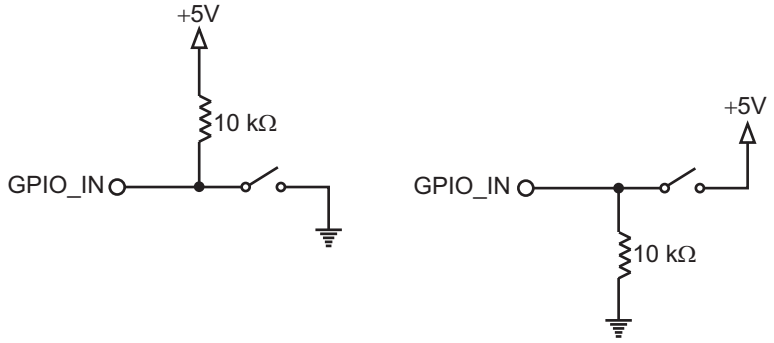
- Open Terminal.
- Launch IDEL by typing - sudo idle.
- After the IDLE is launched, open a new window by FILE → OPEN or Ctrl+N.
- Type the below code in that window:

```
import time
import RPi.GPIO as GPIO       # Import GPIO library
GPIO.setmode(GPIO.BOARD)      # Use board pin numbering
GPIO.setup(25, GPIO.OUT)      # Setup GPIO Pin 25 to OUT
while True:
  GPIO.output(25,True)  # Turn ON LED
  time.sleep(1)         # Wait for one second
  GPIO.output(25,False) # Turn OFF LED
  time.sleep(1)         # Wait for one second
```

- Save the code by FILE → SAVE or Ctrl+S.
- RUN the code RUN → RUN or Ctrl+F5.
- Once the program will start executing, the LED will start blinking.

4.3 Interfacing of a Switch to Raspberry Pi

- The switch or push button can be connected to Raspberry Pi in very simple way. In this case, GPIO pin where the switch is connected has to be configured as an input pin so that it will read the incoming data from the switch. One end of the switch will be connected to GPIO pin through a pull-up or pull-down resistance as shown in below Fig. 4.5.



(a) Input Pull-High **(b) Input Pull-Low**

Fig. 4.5 : Connection of a switch

- As shown in above Fig. 4.5, if an input pin is pull-down then it will read 'Low' when the switch is pressed. In pull-up resistance, it will read 'High' when the switch is pressed.
- Raspberry Pi has an internal pull-up and pull-down resistors, which can be enabled through software. External pull-up or pull-down resistors also can be used.
- Below Fig. 4.6 shows the interfacing of switch to Raspberry Pi.

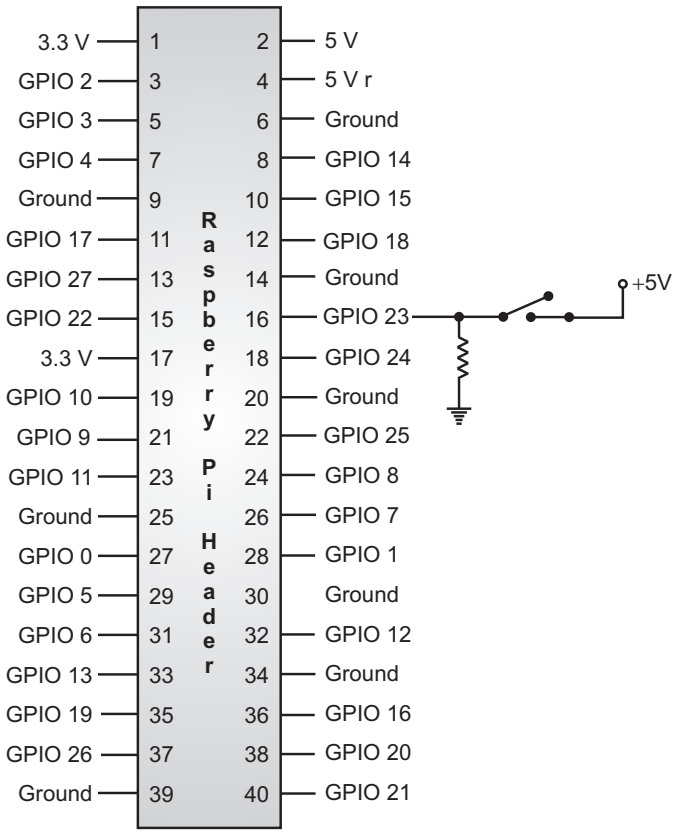


Fig. 4.6: Interfacing a switch to Raspberry Pi

4.3.1 Algorithm for Interfacing a Switch

- Connect a switch as shown in Fig. 4.6.
- Set pin GPIO23 as an input pin.
- Create a variable called count and store 0 in it.
- Save the value of pin GPIO23 into input value.
- Check that value continuously. If the switch is pressed, then the value is true.
- If it is true then increment the counter.
- Print the text to the terminal.
- Execute the program:

```
  import RPi.GPIO as GPIO                              #Import GPIO library
  import time                                          #Import time library
  GPIO.setmode(GPIO.BOARD)                             #Set GPIO pin numbering
  GPIO.setup(23, GPIO.IN, pull_up_down=GPIO.PUD_UP)    #Enable input and pull down resistors
while True:
  input_value = GPIO.input(23)                 #Read and store value of input to a variable
  if input_value == Ture:                      #Check whether switch is pressed
 print('Switch is Pressed')                    #Print 'Switch is Pressed'
 time.sleep(0.5)                               #Delay of 0.5s
```

- The procedure to run the program is same as explained in LED interfacing section.
- The program will print "Switch is Pressed" message when the push switch is pressed. The *time.sleep(0.5)* command will insert necessary delay between two consecutive switch press.
- The LED or any other device can be connected to see the operation of a switch. If switch is ON, then that device will turned ON.

4.4 Interfacing LCD to Raspberry Pi

- LCD is used to display the alphanumeric characters. LCD can be connected to GPIO pins of Raspberry Pi and through program any characters can be displayed on LCD.
- Out of 40 pins of Raspberry Pi, 17 GPIO pins can be used to interface LCD.
- There are +5 V (Pin 2 and 4) and +3.3 V (pin 1 and 17) power output pins are available on Raspberry Pi board.
- LCD requires +5 V, so provide +5 V to 16 × 2 LCD through +5 V power rail.
- Fig. 4.7 shows the interfacing of LCD to Raspberry Pi.

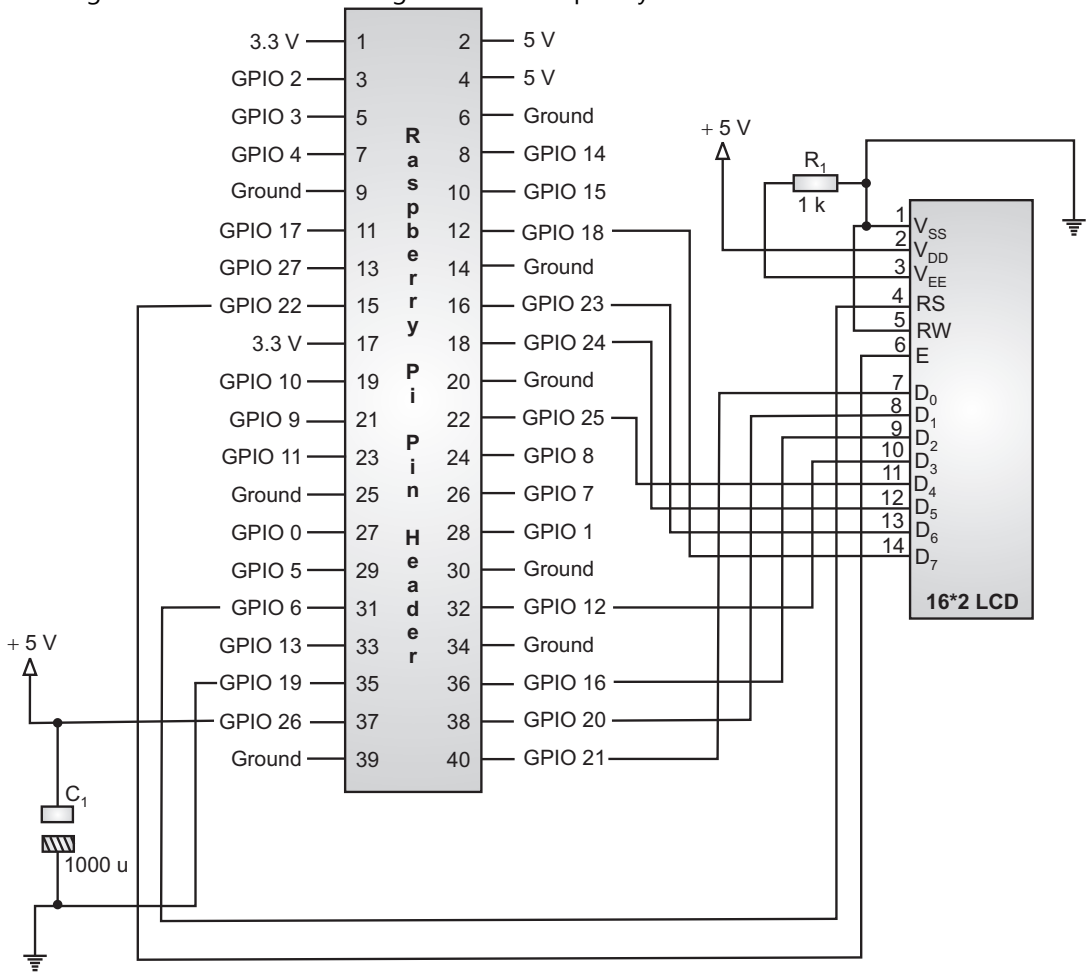


Fig. 4.7: Interfacing LCD to Raspberry Pi

- As shown in above Fig. 4.7, LCD is interfaced to 10 GPIO pins of Raspberry Pi.
- GPIO pin 21, 20, 16, 12, 25, 24, 23 and 18 are used as 'BYTE' and created 'PORT' function to send the data to LCD. GPIO 21 is LSB and GPIO 18 is MSB.
- 16×2 LCD module has 16 pins as tabulated in below Table 4.2.

Table 4.2: Pin Configuration of LCD

Category	Pin No.	Pin Name	Function
Power Pins	1	V_{SS}	Ground pin, connected to Ground.
	2	V_{DD} or V_{CC}	Voltage pin +5 V.
Contrast Pin	3	V_0 or V_{EE}	Contrast setting, connected to V_{CC} thorough a variable resistor.
Control Pins	4	RS	Register Select pin, RS = 0 Command mode, RS = 1 Data mode.
	5	RW	Read/ Write pin, RW = 0 Write mode, RW = 1 Read mode.
	6	E	Enable, a high to low pulse need to enable the LCD.
Data Pins	7-14	D_0- D_7	Data pins stores the data to be displayed on LCD or the command instructions.
Backlight Pins	15	LED+ or A	To power the Backlight +5 V.
	16	LED− or K	Backlight Ground.

4.4.1 Procedure of Sending Data to LCD

- Enable the module by setting E high and RS low to enable commands to LCD.
- Clear screen by sending value 0×01 to data port as a command.
- Provide ASCII code of characters to be displayed.
- Once E pin goes low, the LCD process the received data and display it. So E will be set high before sending the data and then low after sending the data.
- The characters will be sent one by one serially.

The Python program for displaying characters on LCD is as under:

```
import RPi.GPIO as IO      # calling for header file which helps us use GPIO's of PI
import time                # calling for time to provide delays in program
import sys
IO.setmode (IO.BCM)        # programming the GPIO by BCM pin numbers. (like PIN29 as'GPIO5')
IO.setup(6,IO.OUT)         # initialize GPIO Pins as outputs
IO.setup(22,IO.OUT)
IO.setup(21,IO.OUT)
IO.setup(20,IO.OUT)
IO.setup(16,IO.OUT)
```

```
IO.setup(12,IO.OUT)
IO.setup(25,IO.OUT)
IO.setup(24,IO.OUT)
IO.setup(23,IO.OUT)
IO.setup(18,IO.OUT)
def send_a_command (command):    # execute the loop when “sead_a_command” is called
                                 pin=command
    PORT(pin);                   # calling 'PORT' to assign value to data port
    IO.output(6,0)               # putting 0 in RS to tell LCD we are sending command
    IO.output(22,1)              # telling LCD to receive command/data at the port by pulling EN
                                  pin high
    time.sleep(0.05)
    IO.output(22,0)               # pulling down EN pin to tell LCD we have sent the data
    pin=0
    PORT(pin);                    # pulling down the port to stop transmitting
def send_a_character (character):     # execute the loop when “send_a_character” is called
    pin=character
    PORT(pin);
    IO.output(6,1)
    IO.output(22,1)
    time.sleep(0.05)
    IO.output(22,0)
    pin=0
    PORT(pin);
def PORT(pin):                    # assigning PIN by taking PORT value
    if(pin&0×01 == 0×01):
        IO.output(21,1)           # if  bit0 of 8bit 'pin' is true, pull PIN21 high
    else:
        IO.output(21,0)           # if  bit0 of 8bit 'pin' is false, pull PIN21 low
    if(pin&0×02 == 0×02):
        IO.output(20,1)           # if  bit1 of 8bit 'pin' is true, pull PIN20 high
    else:
        IO.output(20,0)           # if  bit1 of 8bit 'pin' is true, pull PIN20 low
    if(pin&0×04 == 0×04):
        IO.output(16,1)
    else:
        IO.output(16,0)
    if(pin&0×08 == 0×08):
        IO.output(12,1)
    else:
        IO.output(12,0)
if(pin&0×10 == 0×10):
        IO.output(25,1)
```

```
    else:
        IO.output(25,0)
    if(pin&0×20 == 0×20):
        IO.output(24,1)
    else:
        IO.output(24,0)
    if(pin&0×40 == 0×40):
        IO.output(23,1)
    else:
        IO.output(23,0)
    if(pin&0×80 == 0×80):
        IO.output(18,1)              # if  bit7 of 8bit 'pin' is true pull PIN18 high
    else:
        IO.output(18,0)              #if  bit7 of 8bit 'pin' is false pull PIN18 low

while 1:
    send_a_command(0×01);            # sending 'all clear' command
    send_a_command(0×38);            # 16*2 line LCD
    send_a_command(0×0E);            # screen and cursor ON
    send_a_character(0×48);          # ASCII code for 'H'
    send_a_character(0×45);          # ASCII code for 'E'
    send_a_character(0×4C);          # ASCII code for 'L'
    send_a_character(0×4C);          # ASCII code for 'L'
    send_a_character(0×4F);          # ASCII code for 'O'

    time.sleep(1)
```

After running the program, 'HELLO' message will be displayed on LCD.

4.5 Setting Bluetooth, WI-FI and Ethernet on Raspberry Pi

- Most of the Raspberry Pi models have on board connectivity options for Wi-Fi and Bluetooth.
- The Raspberry Pi 3, 3B+, Raspberry Pi Zero w and Raspberry Pi 4 have built-in Wi-Fi and Bluetooth facility.
- The Raspberry Pi 3 is the first version of the computer to have in-built wireless and Bluetooth features. Subsequently, all latest models of Raspberry Pi have these features.
- The Raspberry Pi can be easily connected to a Smartphone, TV or any other device through Wi-Fi or Bluetooth.

4.5.1 Setting Wi-Fi on Raspberry Pi

- The Raspberry Pi can be connected to wireless network using desktop tool. For this, a keyboard, mouse and monitor/display have to be connected to Raspberry Pi.
- Before setting Wi-Fi on Raspberry Pi, Ethernet connection has to be disconnected.

- To connect the router, right-click on wireless networking icon on the right corner of the panel. Select the option 'Turn on Wi-Fi', then select the desired network from the menu.
- Input the 'Pre shared Key' when asked and then wait for the connection to be established.
- Wireless network can be set up on the command line. This option is used if Raspberry is accessing using SSH which used for Ethernet.
- To set wireless connection, there are two options; it can be set up through GUI or can be set up on the command line. Below commands will upgrade the Raspberry Pi.

```
sudo apt update
sudo apt upgrade
```

If SSID name is already defined then use it or if it is not defined, use below command;

```
sudo iwlist wlan0 scan
```

This will reveal the SSID in the line "ESSID". Next, open wpa_supplicant.conf:

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Following code has to be added:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=India
network={
ssid="SSID"
psk="PASSWORD"
key_mgmt=WPA-PSK
}
```

Add the SSID and password in the network.

The command **Ctrl + X** is used to exit and save, **Y** and **Enter** used to confirm it. Wireless connectivity should be established immediately. If it is not established, use below command to restart wireless:

```
sudo ifdown wlan0
sudo ifup wlan0
```

Then enter below command:

```
sudo reboot
```

- After rebooting, wireless connection will get established.
- Before booting, Wi-Fi can be set up on Raspberry Pi 3. This is another option to set up Wi-Fi on Raspberry Pi which is described below:
  - Insert the microSD card in Pc's card reader and enter command- /**boot/directory**.
  - Create a text file **wpa_supplicant.conf**, then open it and add the details.
  - Save it, close the file and eject the microSD card.
- Above steps works only with pre-Raspbian Buster OS and various other operating system. Raspbian Buster has a Wi-Fi driver which prevents the use of **wpa_supplicant.conf file.**

- Wi-Fi can be set up using Graphical Interface or GUI. This process is more effective than setting Wi-Fi using command line interface which is described in above section.
- Before starting the Wi-Fi setting, make sure that latest version of Raspberry Pi OS is installed. Follow the below steps:
  - On desktop, locate network icon in the right-hand side and click on the icon to see the list of available Wi-Fi networks.
  - Select available Wi-Fi SSID in the drop-down list.
  - Enter Wi-Fi password into the text box.
  - Click on 'Ok'. Wi-Fi will get connected to Raspberry Pi.
  - Signal strength will be displayed in the upper task bar on the right side.
  - Test connection by opening the web browser.
- Thus, Wi-Fi can be connected to Raspberry Pi.

4.5.2 Setting Bluetooth on Raspberry Pi

- Similar to Wi-Fi, Bluetooth is built-in into Raspbian Buster. For older models of Raspberry Pi, run update and upgrade then use below command:

  ```
  sudo apt install bluetooth-pi
  ```
- Before establishing communication between Raspberry Pi and a Bluetooth enabled device, both the devices should be paired. Pairing a Bluetooth device on Raspberry is similar to pairing Bluetooth on Smartphone or Laptop.
- Bluetooth can be activated from the command line as:

  ```
  bluetoothctl
  ```
- Many options are available with this. Type "help" to see them.
- Bluetooth needs to be enabled, discoverable and made capable of discovering devices. It can be done using below command:

  ```
  power on
  agent on
  scan on
  ```
- With these commands, Raspberry Pi will detect the device, then add the device. The connection can be made by entering connect followed by the MAC address. If password is required on the remote device then enter it when asked.
- Thus, Bluetooth connection will get established.
- For communicating Raspberry Pi and a device via Bluetooth, there are two ways as:
  - Bluetooth GUI (Graphical User Interface).
  - Command Line Interface.
- Before starting communication using Bluetooth, following packages have to be installed:

  Blueman: Blueman is a full featured Bluetooth manager. It provides GUI based setting **panel** Bluetooth manager.

Bluez: Bluez provides the Bluetooth protocol stack and the bluetoothctl utility.

Bluetooth: This package provides all the plugins supported by BluezBluetooth stack.

- Above packages can be installed using following command,

```
sudo apt-get install bluemanbluez Bluetooth
```

- Reboot Raspberry Pi after successful installation of above packages.

4.5.3 Setting Ethernet on Raspberry Pi

- A direct Ethernet connection is always much faster than a Wi-Fi connection to access Internet.
- Raspberry Pi can be directly connected to Ethernet on laptop or desktop with an Ethernet cable bypassing the local network so that bandwidth is not shared with other computers.
- Ethernet can be set up on Raspberry Pi by following steps:
  - Connect Raspberry Pi using PC Ethernet Port using an Ethernet cable.
  - Open Windows, click on menu and select icon for settings.
  - Click on 'Network & Internet'. If Ethernet is already plugged, 'Ethernet' tab is seen on the left side. Click on tab, 'Network and Sharing Center' can be seen on right side of 'Related Settings' section. Click on it.
  - Wi-Fi sharing settings can be modified in Wi-Fi status window. On 'Network and Sharing Center', a connection field for Wi-Fi, click the blue text. It will open 'Wi-Fi Status' window where Wi-Fi status can be modified.
  - In 'Wi-Fi Status' window, click on the 'Properties', it will open 'Wi-Fi Properties'.
  - In 'Wi-Fi Properties' window, click on 'Sharing' tab and check the box for 'Allow other network users to connect through this computer's Internet connection'. In drop-down menu, select Ethernet connection.
  - Check Ethernet connection properties in the 'Network and Sharing Center' window.
  - Check IPv4 settings in 'Ethernet Properties' window. Once 'Use the following IP address' is selected, 'IP address', 'Subnet Mask' and 'Default Gateway' field are in enabled position. Raspberry Pi can be pinged via its hostname such as:

    `ping raspberrypi.local`.
  - After this Ethernet is enabled on Raspberry Pi.
- The Ethernet port of Raspberry Pi can be also configured using Static IP address. Procedure for setting Ethernet on Raspberry Pi is as follows:
  - Review current network settings by typing 'ifconfig' command on command prompt
  - Backup the current network configuration by using below command.

    `sudo cp /etc/dhcpcd.conf /etc/dhcdcp.backup`
  - To change the network setting, edit dhcpcd.conf file to set the static IP address.
  - The following command is used to load the file into an editor

    `sudo nano /etc/dhcpcd.conf`

Place below lines at the top of the file to set Ethernet IP address:

```
interface eth0
static ip_address=10.11.44.124/24
static routers=10.11.44.1
static domain_name_servers=172.16.33.85
```

- Set the IP address , set the routers value to the gateway address. Once the file has been updated, use ctrl to save and exit.
  - Restart the Raspberry Pi by using following command:

    ```
    sudo reboot
    ```
  - Test the new network setup by using 'ping' command and check whether Ethernet cable is connected properly, IP address, mask and gateway are correct.
- Once everything is done properly, Ethernet is enabled.

4.6 Interfacing of Camera to Raspberry Pi

- The camera module is provided with Raspberry Pi which can be used to take pictures and videos.
- Raspberry Pi camera v1.3 have features as: resolution 5 MP, HD Video recording, it can capture wide, still images of resolution 2592 × 1944 pixels and it can be connected through CSI interface.
- Raspberry Pi board has CSI (Camera Serial Interface) port through which camera can be directly connected to Raspberry Pi using 15-pin ribbon cable.
- Camera module should be handled very carefully as camera can be damaged by static electric charge.
- Camera can be installed on Raspberry Pi by inserting the cable into the Raspberry Pi camera port. The camera cable slot is in between the USB and micro-HDMI port on Raspberry Pi board as shown in below Fig. 4.8.

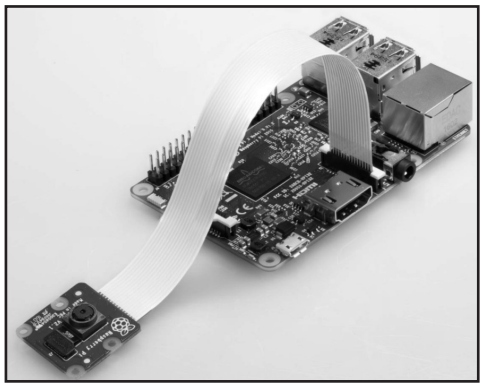


Fig. 4.8 : Interfacing camera module to Raspberry Pi

- There are two versions of camera module: standard version which is designed to take pictures in normal light and NoIR version which can use infrared light source to take pictures in the dark.

Steps to Connect and Install Camera on Raspberry Pi:

- Turn OFF the Raspberry Pi.
- Locate the camera module port on Raspberry Pi and connect camera to it using cable.
- Turn ON Raspberry Pi board.
- Go to main menu and open Raspberry Pi configuration tool, select Interface tab and enable the camera interface.
- Reboot Raspberry Pi board.
- Now, camera is connected to Raspberry Pi. To capture an image in jpeg format, type following command on terminal window:

  ```
  raspistill-o image.jpg
  ```

  'image' is the name of the captured picture which will be stored in Raspberry Pi.
- To capture 10 second video, type following command:

  ```
  raspivid-o video.h264 -t 10000
  ```

  Here, 'video' is the name of the captured video and "10000" is the number of milliseconds.
- Python *picamera* library control the camera module. In IDE, following code will control the camera functions:

  ```
  from picamera import PiCamera
  from time import sleep
  camera = PiCamera()
  camera.start_preview()
  sleep(5)
  camera.stop_preview()
  ```
- Camera preview can be seen if monitor is connected to Raspberry Pi. The preview can be rotated by following commands :

  ```
  camera = PiCamera()
  camera.rotation = 180
  ```

  Image can be rotated by 90, 180 or 270 degrees.
- To take capture still pictures use following code :

  ```
  camera.start_preview()
  sleep(5)
  camera.capture('/home/pi/Desktop/image.jpg')
  camera.stop_preview()
  ```
- As camera will take few seconds to sense the light levels, keep camera in sleep mode for at least two seconds before capturing an image.

- New image is stored on desktop once the program is Run.
- To capture 5 images in a row, type following code :

```
camera.start_preview()
for i in range(5):
   sleep(5)
   camera.capture('/home/pi/Desktop/image%s.jpg' % i)
camera.stop_preview()
```

- To capture the video, type following code :

```
camera.start_preview()
camera.start_recording('/home/pi/Desktop/video.h264')
sleep(5)
camera.stop_recording()
camera.stop_preview()
```

  After running the code, preview of captured video will be displayed on screen.

- Simple code for basic camera is given below :

```
from SimpleCV import Camera, Display
from time import sleep
mycamera = Camera(prop_set={'width' : 320, 'height' : 240})myDisplay = Display(resolution=(320, 240))
while not myDisplay.isDone();
myCamera.getImage().save(myDispaly)
sleep(.1)
```

  Here size of window is 320×240, the code will be in the loop until it get a frame from the camera and display it in the window.

- The main features of an image such as brightness, contrast, resolution can be controlled.
- Image effect can be applied by using- `camera.image_effect`. There are various image effect options available, such as none, solarize, negative, denoise, hatch, film, blur etc.

Many things are possible with camera module such as add buttons to control the camera with the help of GPIO python code, integrate camera with Minecraft Pi, post the camera pictures to Twitter automatically.

4.7 Interfacing GSM Module to Raspberry Pi

- Dial up MODEMs are very familiar and widely used for land line telephone network. However, it has limitations when used in embedded systems.
- GSM (Global System for Mobile) is introduced to rectify the limitations of dial-up MODEM which is based on SIM card.
- GSM is almost similar to mobile communication system. Data can be transmitted and received by any embedded system through GSM module.

- GSM module can be interfaced to Raspberry Pi and can be used in image, video, audio processing, automation, surveillance.
- GSM module can be interface to Raspberry Pi in two ways: through USB or through GPIO interfacing.
- A micro USB cable can be used for connecting Raspberry Pi to GSM module which provide power and establish a serial communication.
- Below Fig. 4.9 shows interfacing of Raspberry Pi and GSM module through micro USB cable.

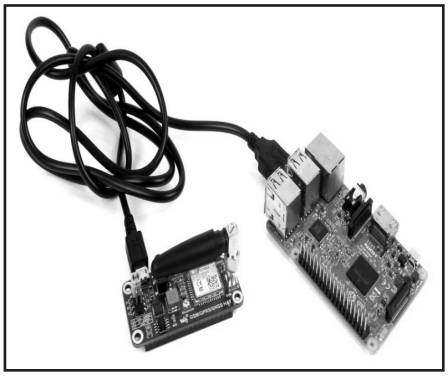


Fig. 4.9: Interfacing GSM module to Raspberry pi using USB cable

Source: https://www.rhydolabz.com/wiki/?p=18639#:~:text=Power%20on%20the%20raspberry%20pi, be%20in%20an%20on%20state.

- SIM 868 and SIM 900 GSM modules are almost same and have similar features. SIM 868 modules have Quad-Band GSM/GPRS and can be used for satellite navigation.
- On SIM 868 module, there is yellow jumper which is shorted to make three terminal pairs as:

  A: Control the SIM 868 through USB to UART.

  B: Control the SIM 868 through Raspberry Pi.

  C: Access Raspberry Pi through USB to UART.
- GSM module can be controlled using UART, I2C or SPI interfaces, if it is serially connected to Raspberry Pi.
- GSM module understands AT commands which can be passed through UART, I2C or SPI protocol. GSM module can be checked, configured and make ready to receive voice calls, SMS, MMS etc. with the help of these AT commands.

Steps for Interfacing GSM Module to Raspberry Pi :

- Insert a SIM card into GSM modem and connect it to Raspberry Pi using USB cable as shown in Fig. 4.9.
- Power ON the Raspberry Pi and Rasbian OS.
- Make sure that, transmission and reception pins and ground pins are properly connected.
- Power ON the GSM module and wait for few seconds for SIM initialization.

- Run below Python program for sending a message through Raspberry Pi and GSM module.
- In the program, proper libraries are imported and serial communication is enabled. Modem is controlled through AT command.

[I] Program for sending message using GSM module

```
# Code for sending message using GSM module
import RPi.GPIO as GPIO
import os, time
 GPIO.setmode(GPIO.BOARD)
 # Enable Serial Communication
port = serial.Serial("/dev/ttyUSB0", baudrate=9600, timeout=1)
 # Transmitting AT Commands to the Modem
# '\r\n' indicates the Enter key
port.write('AT'+'\r\n')
rcv = port.read(10)
print rcv
time.sleep(.1)
   port.write('ATE0'+'\r\n')       # Disable the Echo
rcv = port.read(10)
print rcv
time.sleep(.1)
 port.write('AT+CMGF=1'+'\r\n')  # Select Message format as Text mode
rcv = port.read(10)
print rcv
time.sleep(.1)
 port.write('AT+CNMI=2,1,0,0,0'+'\r\n')   # New SMS Message Indications
rcv = port.read(10)
print rcv
time.sleep(.1)
 # Sending a message to a particular Number
port.write('AT+CMGS="9422321170"'+'\r\n')
rcv = port.read(10)
print rcv
time.sleep(.1)
 port.write('Hello'+'\r\n')  # Message
rcv = port.read(10)
print rcv
```

```
 port.write("\x1A") # Enable to send SMS
for i in range(10):
    rcv = port.read(10)
    print rcv
```

After executing the above program, the message will be sent to mobile number specified in the program code.

[II] Program for sending message using GSM module

```
# Code for receiving SMS via GSM module
import RPi.GPIO as GPIO
import os, time
 # Find a suitable character in a text or string and get its position
def locate(str, ch):
    for i, ltr in enumerate(str):
        if ltr == ch:
            yield i
 GPIO.setmode(GPIO.BOARD)
 # Enable Serial Communication
port = serial.Serial("/dev/ttyAMA0", baudrate=9600, timeout=1)
 # Transmitting AT Commands to the Modem
# '\r\n' indicates the Enter key
 port.write('AT'+'\r\n')
port.write("\x0D\x0A")
rcv = port.read(10)
print rcv
time.sleep(1)
 port.write('ATE0'+'\r\n')                # Disable the Echo
rcv = port.read(10)
print rcv
time.sleep(1)
 port.write('AT+CMGF=1'+'\r\n')           #  Select  Message  format  as  Text
mode
rcv = port.read(10)
print rcv
time.sleep(1)
 port.write('AT+CNMI=2,1,0,0,0'+'\r\n')    # New SMS Message Indications
rcv = port.read(10)
print rcv
```

```
time.sleep(1)
ck=1
while ck==1:
    rcv = port.read(10)
    print rcv
    fd=rcv
    if len(rcv)&gt;3:                    # check if any data received
        ck=12
        for i in range(5):
            rcv = port.read(10)
            print rcv
            fd=fd+rcv                 # Extract the complete data
 # Extract the message number shown in between the characters "," and '\r'
         p=list(locate(fd, ","))
        q=list(locate(fd, '\r'))
        MsgNo=fd[p[0]+1:q[1]]
 # Read the message corresponds to the message number
        rd=port.write('AT+CMGR='+MsgNo+'\r\n')
        msg=''
        for j in range(10):
            rcv = port.read(20)
            msg=msg+rcv
        print msg
    time.sleep(0.1)
```

After executing the above program, the message will be received and retrieved on the mobile number specified in the program code.

4.8 Interfacing Ultrasonic Sensor to Raspberry Pi

- Ultrasonic sensors are used to measure the distance between two objects using ultrasonic waves. Ultrasonic waves are used as they are more accurate and not audible to human ear.
- HC-SR04 ultrasonic sensor as shown in below Fig. 4.10 is widely used for distance measurement up to 400 cm. It consists of transmitter, receiver and control circuit. As shown in Fig. 4.10, it has V_{cc}, Trig, Echo and GND.
- The ultrasonic transmitter transmits waves (of 40 kHz) get reflected by the target which are received by the receiver.
- The Echo pin will change the state and it becomes high. It remains in high state until it hit the target object and returns to receiver.

- Based on the time for which Echo pin stays high, distance can be calculated using the formula: **Speed = Distance/Time**.

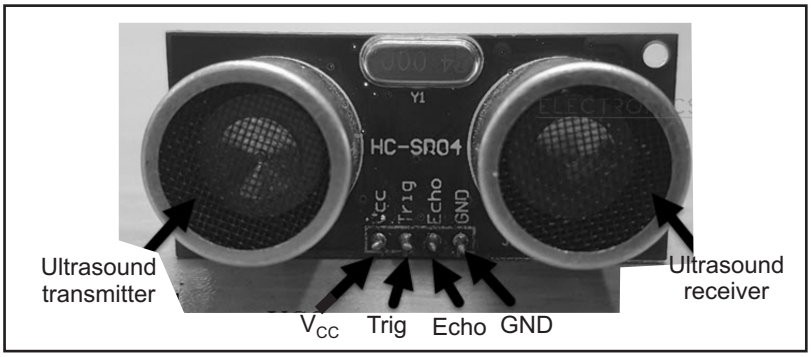


Fig. 4.10 : Ultrasonic sensor HC – SR04

- The ultrasonic sensor is interfaced to Raspberry Pi as shown in below Fig. 4.11. As Raspberry Pi works on 3.3V and ultrasonic sensor require 5 V, a level converter is connected as shown in Fig. 4.11.

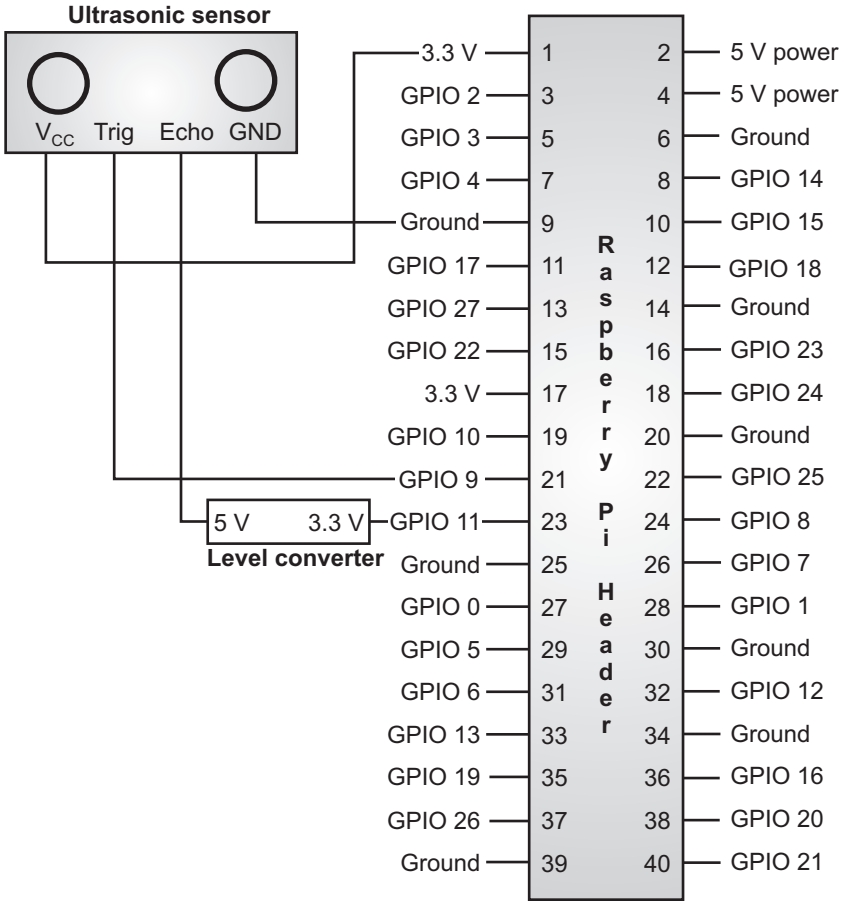


Fig. 4.11 : Interfacing ultrasonic sensor to Raspberry Pi

- The Raspberry Pi read the Echo pin to calculate the time, so this pin is connected to GPIO and configured as an input pin. The output is available on Trig pin, so it is configured as output pin.
- The range of ultrasonic sensor is from 2 cm to 400 cm. So in the program it is checked whether the object is within the range or not.

The python program for interfacing ultrasonic sensor is given below:

```
import RPi.GPIO as GPIO                  #Import GPIO library
import time                              #Import time library
GPIO.setmode(GPIO.BCM)                   #Set GPIO pin numbering
TRIG = 21                                #connect pin 21 to TRIG
ECHO = 23                                #connect pin 23 to ECHO
print "Distance measurement starts"
GPIO.setup(TRIG,GPIO.OUT)                   #configure Trig as GPIO out
GPIO.setup(ECHO,GPIO.IN)                    #configure Echo as GPIO in
while True:
  GPIO.output(TRIG, False)                  #Set TRIG as LOW
  print "Waitng For Sensor To Settle"
  time.sleep(2)                             #Delay of 2 seconds
  GPIO.output(TRIG, True)                  #Set TRIG as HIGH
  time.sleep(0.00001)                       #Delay of 0.00001 seconds
  GPIO.output(TRIG, False)                 #Set TRIG as LOW
  while GPIO.input(ECHO)==0:               #Check whether the ECHO is LOW
    pulse_start = time.time()              #Saves the last known time of LOW pulse
  while GPIO.input(ECHO)==1:               #Check whether the ECHO is HIGH
    pulse_end = time.time()                #Saves the last known time of HIGH pulse
  pulse_duration = pulse_end - pulse_start #Get pulse duration to a variable
  distance = pulse_duration * 17150  #Multiply pulse duration by 17150 to get distance
  distance = round(distance, 2)            #Round to two decimal points
  if distance > 2 and distance < 400:      #Check whether the distance is within range
    print "Distance:",distance - 0.5,"cm"  #Print distance with 0.5 cm calibration
  else:
    print "Object is Out Of Range"         #display object is out of range
time.sleep(0.00001)
```

After running the program, corresponding message will get displayed.

4.9 Interfacing PIR Sensor to Raspberry Pi

- Passive Infrared (PIR) sensor detects any movement of human or animals which emits infrared.
- PIR sensor detects change in the infrared radiation. PIR sensor is shown in below Fig. 4.12.

- When any object or human is detected in the field of view of PIR sensor, the temperature of PIR sensor rises and again come to ambient temperature. PIR sensor converts this temperature change into the change in output which is considered as movement detection.
- The output of PIR sensor is high (5 V) when it senses the motion and low (0 V) when there is no motion. Therefore, PIR sensor is also known as Motion Sensor.

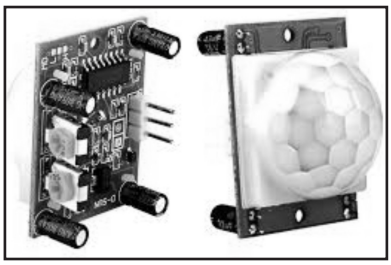


Fig. 4.12: PIR sensor

- PIR sensors are used in many applications such as lighting control system, home automation, security systems etc.
- When the movement is detected by the PIR sensor, its output goes high which will be read by Raspberry Pi and LED can be turned ON as an indication of movement detection. So, when there is a movement in front of PIR sensor, LED will be ON.
- PIR is an input to Raspberry Pi and LED is treated as output. So PIR is configured as input pin and LED is configured as output pin of Raspberry Pi.
- PIR sensor is interfaced to Raspberry Pi as shown in below Fig. 4.13.

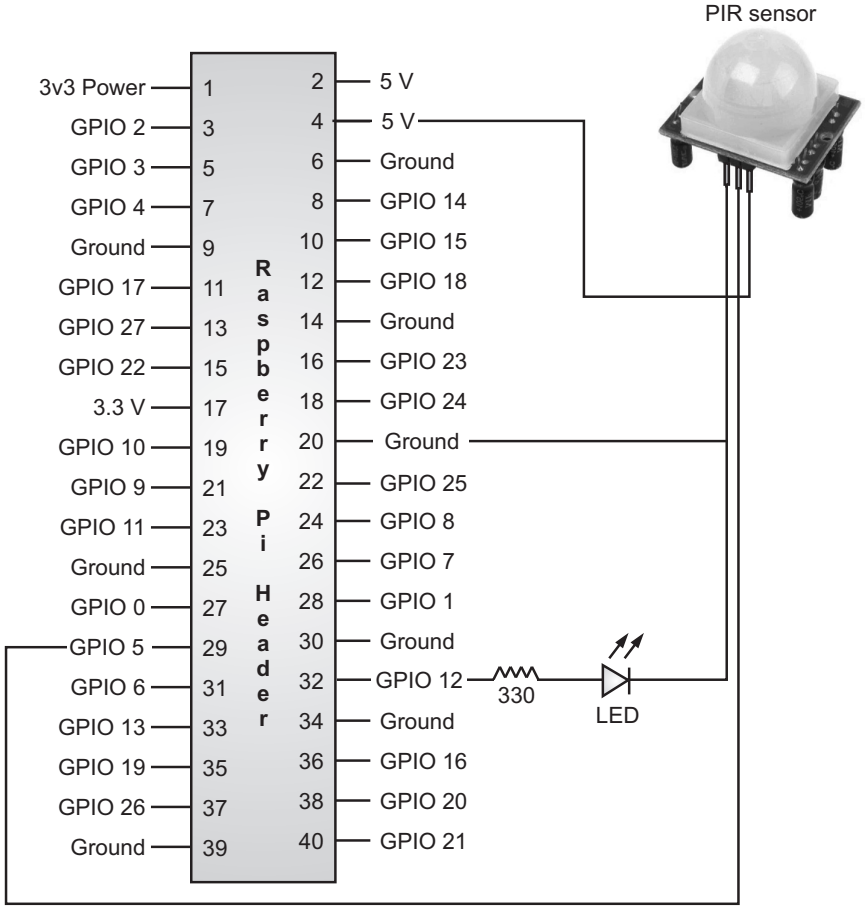


Fig. 4.13 : Interfacing PIR sensor to Raspberry Pi

- The program code for interfacing PIR sensor is given below:

```
import RPi.GPIO as GPIO

PIR_input = 29                      #read input from PIR
LED = 32                            #LED for indication of movement detection
GPIO.setmode(GPIO.BOARD)            #choose pin no. system
GPIO.setup(PIR_input, GPIO.IN)
GPIO.setup(LED, GPIO.OUT)
GPIO.output(LED, GPIO.LOW)
while True:                         #when motion detected turn LED ON
    if(GPIO.input(PIR_input)):
        GPIO.output(LED, GPIO.HIGH)
    else:
        GPIO.output(LED, GPIO.LOW)
```

- Instead of LED, buzzer can be connected which will buzz when the movement is detected by the PIR sensor.

4.10 Interfacing Finger Print Reader to Raspberry Pi

- As fingerprints are unique and no two fingerprints can match, so it is always used in security systems.
- Fingerprint technology is most popular technology to verify the identity of an individual.
- The characteristics such as ridges or minutia patterns of a Fingerprint found within print area are used for matching purpose.
- The fingerprint sensing process consists of capturing the image, extracting the features of fingerprint, storing its digital data and then comparing the current image with the stored fingerprint image.
- An electronic device which records a digital image of the fingerprint pattern is known as fingerprint reader or scanner.
- Fingerprint reader provides high level security, so it has many advantages as compared with other security systems.
- Fingerprint sensor R307 is widely used to read the fingerprint in biometrics systems. It can store 1000 fingerprint images. It is operated on 3.3 V and can be interfaced through USB. Fig. 4.14 shows R307 fingerprint sensor.

Fig. 4.14: Fingerprint sensor R307

- To interface fingerprint reader to Raspberry Pi, USB to TTL converter is required which can support various baud rates for serial communication. CP2102 USB to TTL converter is widely used in between fingerprint reader and Raspberry Pi.

- Interfacing of fingerprint reader is shown in below Fig. 4.15.

Fig. 4.15

- To write a python program, first fingerprint package has to be imported from python library. Then the sensor is initialised with appropriate baud rate and checked for its password. The fingerprints are stored in memory.
- At the beginning of the program, packages required to read fingerprint has to be downloaded.

The steps for interfacing the fingerprint reader are given below:

(i) Install required library and packages by typing following commands:

```
sudo bash
wget –O – http://apt.pm-codeworks.de/pm-codeworks.de.gpg | apt-key add –
wget http://apt.pm-codeworks.de/pm-codeworks.list -P /etc/apt/sources.list.d/
```

(ii) Update library and download fingerprint sensor library by typing following commands:

```
sudo apt-get update
sudo apt-get install python-fingerprint –yes
```

(iii) Check USB port where fingerprint sensor is connected by using below command:

```
ls /dev/ttyUSB*
```

(iv) Allot memory space to store fingerprint image in digital form.

(v) Below code will initialize fingerprint sensor and allot memory space:

```
import time
from pyfingerprint.pyfingerprint import PyFingerprint
# Detect new finger
# Initialize the sensor
init:
    f = PyFingerprint('/dev/ttyUSB0', 9600, 0xFFFFFFFF, 0x00000000)

    if ( f.verifyPassword() == False ):
        raise ValueError('The fingerprint sensor password is wrong!')
except Exception as e:
    print('The fingerprint sensor could not be initialized!')
    print('Exception message: ' + str(e))
    exit(1)
# Gets some sensor information
print('Currently used templates: ' + str(f.getTemplateCount()) +'/'+
str(f.getStorageCapacity()))
```

After executing above steps, the fingerprint is detected and stored in char buffer 1.

(vi) if-else loop can be used to check whether fingerprint is matching with the particular image which is already stored.

```
## Enroll new finger
try:
    print('Waiting for fingerprint...')

    while ( f.readImage() == False ):           #wait for new fingerprint
        pass

   f.convertImage(0x01)
             # Convert image into digital form and stores it in char buffer 1
   result = f.searchTemplate()  # Checks if fingerprint is already enrolled
    positionNumber = result[0]
    if ( positionNumber >= 0 ):
        print('Template already exists at position #' + str(positionNumber))
        exit(0)
    print('Remove finger...')
    time.sleep(2)
    print('Wait for same finger again...')
    while ( f.readImage() == False ):
        pass
```

(vii) The fingerprint is detected and stored in char buffer 2. This process is repeated and if a new fingerprint is detected it is stored in another variable called PositionNumber.

(viii) Python program code is given below:

```
import time
from pyfingerprint.pyfingerprint import PyFingerprint
# initialize the sensor
init:
    f = PyFingerprint('/dev/ttyUSB0', 9600, 0xFFFFFFFF, 0x00000000)

    if ( f.verifyPassword() == False ):
        raise ValueError('The fingerprint password is wrong!')
except Exception as e:
    print('The fingerprint sensor could not be initialized!')
    print('Exception message: ' + str(e))
    exit(1)
```

```
print('Currently used templates:
                      ' + str(f.getTemplateCount()) +'/'+ str(f.getStorageCapacity()))
# enroll new finger
try:
    print('Waiting for fingerprint...')
    while ( f.readImage() == False ):
        pass
    f.convertImage(0x01)
    result = f.searchTemplate()                # Checks if finger is already enrolled
    positionNumber = result[0]
    if ( positionNumber >= 0 ):
        print('Fingerprint image already exists at position #' + str(positionNumber))
        exit(0)
    print('Remove finger...')
    time.sleep(2)
    print('Wait for same finger again...')
    while ( f.readImage() == False ):          # Wait for same fingerprint
        pass
    f.convertImage(0x02)
                     # Convert image into digital form and stores it in charbuffer 2
   if ( f.compareCharacteristics() == 0 ):     # Compare two charbuffers
        raise Exception('Fingers do not match')
    f.createTemplate()                 # Creates a template
    positionNumber = f.storeTemplate()  # Save fingerprint at new position number
    print('Finger enrolled successfully!')
    print('New position #' + str(positionNumber))
except Exception as e:
    print('Operation failed!')
    print('Exception message: ' + str(e))
    exit(1)
```

(ix) Below code will delete registered fingerprint from the stored data:

```
from pyfingerprint.pyfingerprint import PyFingerprint
init:
    f = PyFingerprint('/dev/ttyUSB0', 9600, 0xFFFFFFFF, 0x00000000)
    if ( f.verifyPassword() == False ):
        raise ValueError('The fingerprint password is wrong!')
```

```
except Exception as e:
    print('The fingerprint sensor could not be initialized!')
    print('Exception message: ' + str(e))
    exit(1)
print('Currently used templates:
                        ' + str(f.getTemplateCount()) +'/'+ str(f.getStorageCapacity()))
delet:
       #delete the template of the fingerprint
    positionNumber = input('Please enter the template position you want to delete: ')
    positionNumber = int(positionNumber)
    if ( f.deleteTemplate(positionNumber) == True ):
        print('Template deleted!')
except Exception as e:
    print('Operation failed!')
    print('Exception message: ' + str(e))
    exit(1)
```

(x) Below code is used to search fingerprint from the stored data.

```
import hashlib
from pyfingerprint.pyfingerprint import PyFingerprint
# initialize the sensor
init:
    f = PyFingerprint('/dev/ttyUSB0', 9600, 0xFFFFFFFF, 0x00000000)
    if ( f.verifyPassword() == False ):
        raise ValueError('The fingerprint password is wrong!')
except Exception as e:
    print('The fingerprint sensor could not be initialized!')
    print('Exception message: ' + str(e))
    exit(1)
print('Currently used templates:
                        ' + str(f.getTemplateCount()) +'/'+ str(f.getStorageCapacity()))
search:
    print('Waiting for finger...')         # search the finger and calculate hash
    while ( f.readImage() == False ):      #read fingerprint image
        pass
    f.convertImage(0x01) #Convert image into digital form and stores it in charbuffer 1
    result = f.searchTemplate()
    positionNumber = result[0]
```

```
accuracyScore = result[1]
if ( positionNumber == -1 ):
    print('No match found!')
    exit(0)
else:
    print('Found template at position #' + str(positionNumber))
    print('The accuracy score is: ' + str(accuracyScore))
   exit(1)
```

Think Over It

- Is it possible to do image processing using Raspberry Pi? How?
- How ECG sensor can be interfaced to Raspberry Pi?
- Is it possible to implement MATLAB code on Raspberry Pi?
- Can FPGA board used with Raspberry Pi?

Points to Remember

- Python interpreter can be run in two ways: as an interactive shell to execute individual commands or as a command-line program to execute standalone programs or scripts.
- The integrated development environment (IDE) bundled with Python and the Raspberry Pi is called as IDEL.
- To write a Python program for Raspberry, first Operating System has to be installed and then Raspberry has to be configured. Booting Raspberry Pi for the first time include following steps:
  - Push the microSD card into the socket on the bottom of the board.
  - Plug in a USB keyboard and mouse.
  - Plug the HDMI output into TV or monitor. Make sure that monitor is ON and set to the correct input.
  - Plug in power supply and switch ON the board.
- **In Raspbian, Python and GPIO library are preinstalled.**
- Pin number 3, 5, 7, 8, 10, 11, 12, 13, 15, 16, 18, 19, 21, 22, 23, 24, 26 are the GPIO pins of Raspberry Pi.
- Most of the Raspberry Pi models have on board connectivity options for Wi-Fi and Bluetooth.
- The Raspberry Pi 3, 3B+, Raspberry Pi Zero w and Raspberry Pi 4 have built-in Wi-Fi and Bluetooth facility.

- The Raspberry Pi 3 is the first version of the computer to have in-built wireless and Bluetooth features. Subsequently, all latest models of Raspberry Pi have these features.
- **Blueman:** Blueman is a full featured Bluetooth manager. It provides GUI based setting panel **Bluetooth manager.**
- **Bluez:** Bluez provides the Bluetooth protocol stack and the bluetoothctl utility.
- A direct Ethernet connection is always much faster than a Wi-Fi connection to access Internet.
- Raspberry Pi can be directly connected to Ethernet on laptop or desktop with an Ethernet cable bypassing the local network so that bandwidth is not shared with other computers.
- The camera module is provided with Raspberry Pi which can be used to take pictures and videos.
- Raspberry Pi camera v1.3 have features as: resolution 5 MP, HD Video recording, it can capture wide, still images of resolution 2592 $\times$ 1944 pixels and it can be connected through CSI interface.
- Raspberry Pi board has CSI (Camera Serial Interface) port through which camera can be directly connected to Raspberry Pi using 15-pin ribbon cable.
- Camera module should be handled very carefully as camera can be damaged by static electric charge.
- Ultrasonic sensors are used to measure the distance between two objects using ultrasonic waves. Ultrasonic waves are used as they are more accurate and not audible to human ear.
- HC-SR04 ultrasonic sensor is widely used for distance measurement up to 400 cm. It consists of transmitter, receiver and control circuit. As shown in Fig. 4.10, it has V_{CC}, Trig, Echo and GND.
- Passive Infrared (PIR) sensor detects any movement of human or animals which emits infrared.
- **PIR sensor detects change in the infrared radiation.**
- Fingerprint technology is most popular technology to verify the identity of an individual.
- The characteristics such as ridges or minutia patterns of a Fingerprint found within print area are used for matching purpose.

Exercises

[A] True or False :

1. The GPIO pin number and physical pin number of Raspberry Pi are different.
2. Python is not an interpreted language.
3. The integrated development environment (IDE) bundled with Python and the Raspberry Pi is called as IDEL.
4. If Raspbian Lite is installed then keyboard, mouse and monitor are not required.
5. In Raspbian, Python and GPIO library are not preinstalled.

6. A direct Ethernet connection is always much faster than a Wi-Fi connection to access Internet.
7. The Raspberry Pi 3 is the first version of the computer to have in-built wireless and Bluetooth features.

[B] Multiple Choice Questions :

1. Which operating system does Python supports?
   (a) Windows (b) Linux
   (c) Both (a) and (b) (d) None of these
2. Following models of Raspberry Pi have Bluetooth facility.
   (a) Raspberry Pi 3 (b) Raspberry Pi Zero
   (c) Raspberry Pi 4 (d) All of these
3. PIR sensor detects change in the …… radiation.
   (a) Infrared (b) Micro
   (c) Ultraviolet (d) None of these
4. Ultrasonic sensor widely used to measure ……
   (a) Time (b) Distance
   (c) Frequency (d) None of these
5. Which library/libraries have to be imported to interface fingerprint reader to Raspberry Pi?
   (a) PyFingerprint (b) Numpy
   (c) Both (a) and (b) (d) None of these

[C] Fill in the Blanks:

1. Python is an _____ language.
2. The integrated development environment (IDE) bundled with Python and the Raspberry Pi is called as ____ .
3. The Raspberry Pi 3 is the first version of the computer to have in-built ____ and _____ features.
4. Ultrasonic sensors are used to measure the distance between two objects using ____ waves.
5. PIR sensor detects change in the _____ radiation.
6. In Raspbian, _____ and _____ library are preinstalled.

[D] Short Answer Questions:

1. What is IDEL?
2. List two popular modules used in Raspberry Pi.
3. What is the difference between Raspbian and Raspbian Lite?
4. How many pins are as GPIO in Raspberry Pi?
5. What is Bluetooth technology?
6. What is the difference between Bluetooth and Wi-Fi?
7. What is the use of GSM module?

8. Explain the working principle of ultrasonic sensor.
9. What is the application of PIR sensor?
10. State the applications of fingerprint reader.

[E] Long Answer Questions:

1. Explain the steps for booting the Raspberry Pi.
2. Draw the diagram of interfacing the LED to Raspberry Pi and write python program to blink the LED.
3. Explain the interfacing of a switch to Raspberry with the help of diagram and python program.
4. Draw the diagram of interfacing of LCD to Raspberry Pi. Write python program to display the message on LCD.
5. Explain the procedure of setting the Bluetooth on Raspberry Pi.
6. Explain the procedure of setting Wi-Fi on Raspberry Pi.
7. Explain the procedure of installing the Ethernet on Raspberry Pi.
8. Explain how camera is interfaced to Raspberry Pi? Write python program to capture images using the camera.
9. Draw the diagram of interfacing the GSM module to Raspberry Pi. Write a python program to send SMS on given mobile number.
10. Explain how ultrasonic sensor is interfaced to Raspberry Pi. Write a python program to measure the distance between two objects.
11. Write a python program to detect the movement using PIR sensor to Raspberry Pi.
12. Explain with help of neat diagram the interfacing of fingerprint reader to Raspberry Pi. Write a python program to read and store the fingerprint.

Answers

[A] True or False :

(1) True (2) False (3) True
(4) True (5) False (6) True
(7) True

[B] Multiple Choice Questions :

1. (a)	2. (d)	3. (a)	4. (b)	5. (a)

[C] Fill in the Blanks :

(1) interpreted (2) IDEL (3) wireless and bluetooth
(4) ultrasonic (5) infrared (6) Python and GPIO

Model Question Paper - I

Time : 2 Hours **Maximum Marks : 35**

Note :

1. Q.1 is compulsory.
2. Solve any three questions from Q.2 to Q.5.
3. Q.2 to Q.5. carry equal marks.

Q.1. Answer the flowing questions in one or two sentences. (Any 5) **(5 marks)**

1. What is SBC?
2. Write any one advantage and disadvantage of SoC.
3. What is Raspbian? List features of Raspbian.
4. State any two types of SBC.
5. List two popular modules used in Raspberry Pi.
6. What is the use of GSM module?
7. List at 4 features of Python.

Q.2. Answer the following : **(10 marks)**

[A] 1. Draw and explain the block diagram of single board computer. (3)

2. What is Python? List features of Python. (3)

[B] Draw the diagram of interfacing the LED to Raspberry Pi and write python program to blink the LED. (4)

Q.3. Answer the following : **(10 marks)**

[A] 1. Compare any three types of SBC. (3)

2. Explain membrane keyboard. (3)

[B] Draw and explain the architecture of SoC with the help of neat diagram. (4)

Q.4. Answer the following : **(10 marks)**

[A] 1. Write a short note of comparison of Raspberry Pi modules. (3)

2. Write a short note on Python variables. (3)

[B] Explain how camera is interfaced to Raspberry Pi? Write python program to capture images using the camera. (4)

Q.5. Write a short note on the following : (Any 4) **(10 marks)**

1. Break and Pass statements used in Python.
2. Four features of ARM1176JZ-S.
3. Prefetch unit.
4. Data types used in Python.
5. Pipelining used in Raspberry Pi.
6. Types of keyboard.

Model Question Paper - II

Time : 2 Hours **Maximum Marks : 35**

Note :

1. Q.1 is compulsory.
2. Solve any three questions from Q.2 to Q.5.
3. Q.2 to Q.5. carry equal marks.

Q.1. Answer the flowing questions in one or two sentences. (Any 5) **(5 marks)**

1. What is SoC?
2. Write any one advantage and disadvantage of SBC.
3. What is Python? List features of Python.
4. State any types of keyboard.
5. Write two softwares used for Raspberry Pi.
6. What is the use of ultrasonic sensor?
7. List at 4 features of Raspberry Pi.

Q.2. Answer the following : **(10 marks)**

[A] 1. Draw and explain the block diagram of SoC. (4)

2. Give at least 4 examples of data type conversion functions. (2)

[B] Explain with help of neat diagram the interfacing of fingerprint reader to Raspberry Pi. Write a python program to read and store the fingerprint. (4)

Q.3. Answer the following : **(10 marks)**

[A] 1. What are the advantages and disadvantages of SBC? (3)

2. Explain the procedure of setting Wi-Fi on Raspberry Pi. (3)

[B] Draw the diagram of interfacing the GSM module to Raspberry Pi. Write a python program to send SMS on given mobile number. (4)

Q.4. Answer the following : **(10 marks)**

[A] 1. Explain if else, for and while loops used in Python. (3)

2. Explain CPU pipelining stages. (3)

[B] Explain how ultrasonic sensor is interfaced to Raspberry Pi. Write a python program to measure the distance between two objects. (4)

Q.5. Write a short note on the following : (Any 4) **(10 marks)**

1. LCD display.
2. Branch prediction.
3. Continue and try statements used in Python.
4. Functions used in Python.
5. Procedure of installing the Ethernet on Raspberry Pi.
6. GPU of Raspberry Pi.