1

SOC AND RASPBERRY PI

Unit Structure

1.0 OBJECTIVES

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

- Understand the concept and structure of System on Chip
- Describe and illustrate several SoC products
- Explain and describe about the ARM 8 Architecture
- Introduce and prepare Raspberry Pi with hardware and installation

1.1 INTRODUCTION

System on Chip

A system on chip also known as SoC is an integrated circuit (IC) that integrates all the components into a single chip that is the complete system is present on the same single chip and hence its name. It has analog, digital,

mixed signal and other radio frequency function all present on a single chip substrate. Now-a-days, SoCs applications are more commonly found in electronics industry due to its low power consumption. Also, the greater use of SoCs is found in embedded system applications. SoCs generally consists of control unit (comprises of microprocessor, microcontroller, digital signal processor etc.); memory blocks (i.e ROM, RAM, Flash memory, EEPROM); timing units (oscillators, PLLs); other peripherals (it consists counter timers, real-time timers and power on reset generators); basic SoC interfaces (analog interfaces, external interfaces, voltage regulators and power management units).

Raspberry Pi

Raspberry Pi is a series of compact single-board computers developed by the Raspberry Pi Foundation in collaboration with Broadcom in the United Kingdom. These projects are generally inclined towards teaching and promoting basic computer science in schools and in developing countries. Due to its low cost, modularity and open design it finds wide application ranging from weather monitoring, robotics and many more.

Several generations have been released of Raspberry Pi's such as Raspberry Pi Model B (February 2012), followed by Model A, Model B+ (in 2014), Raspberry Pi 2(February 2015), Raspberry Pi Zero (November 2015), Raspberry Pi Zero W (On 28 February 2017), Raspberry Pi Zero WH (On 12 January 2018), Raspberry Pi 3 Model B (February 2016), Raspberry Pi 3 Model B+ (2018), Raspberry Pi 4 Model B (released in June 2019), Raspberry Pi 400 (November 2020) and Raspberry Pi Pico (in January 2021).

1.2 SYSTEM ON CHIP

1.2.1 What is System on Chip (SoC)?

SoC is an integrated circuit embedded on a small single platform coin-sized chip with a microprocessor / microcontroller along with all other electronic components integrated onto it. As the name suggests it is the entire system (complete circuit) on a single chip. SoC design usually incorporates central processing unit, input and output ports, memory, secondary storage devices, as well as analog input and output blocks, digital or analog signal processing system or a floating-point unit and peripheral interfaces such as I2C, SPI, UART, CAN, Timers, etc. It is capable of performing several tasks including signal processing, wireless communication, artificial intelligence and more.

1.2.1.2 Why SoC?

As technology is becoming more and more advanced, the main motivator and primary goal is to reduce energy waste, save up on spending costs, as well as reduce the space occupied by large systems. This essential requirement is possible by SoC as it size-down multichip design onto a single processor comparatively consuming less power than before. These chips helped us to developed portable devices that can be carried anywhere

and everywhere easily without compromising on the capability and functionality of the gadgets. SoCs have a plethora of practical uses that are both unlimited and priceless. These are associated with systems pertaining to the Internet of Things, embedded systems, smartphones, cameras, tablets, cars, wireless technologies etc.

The working of a SoC can be best described with an example of smartphones. When you use your cell phones you not only make and receive the calls; you LSO use it for browsing the internet, listening audio, watching videos, taking photos, playing games etc. Multiple components, such as a graphics card, internet support, wireless connections, GPS, and several more aspects, make all of these features feasible. All of these components can be merged into a single chip, which can then be shrunk down to fit in the palm of your hand and carried about. In recent years, SoC technology are used in small sized personal computers, laptops for reducing power consumption thus further improving the performance by using a single chip managing all the functionalities of the system.

1.2.1.3 SoC Building Blocks

SoC comprises of several building blocks as shown in the figure 1.1.

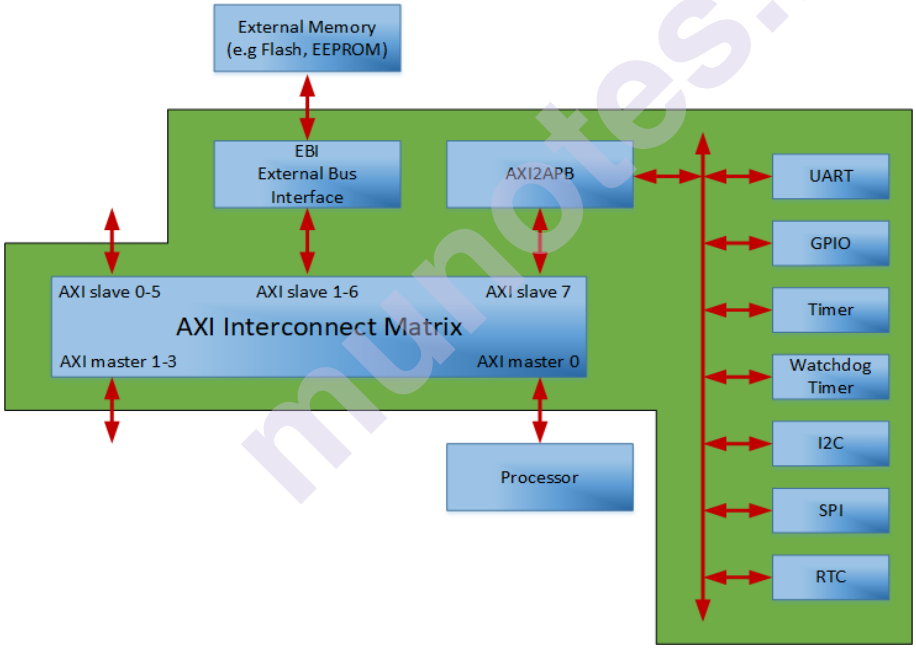


Figure 1.1 Building blocks of SoC

- Firstly, at its core, a system on chip must consist of a processor that will define all its functions. Generally, an SoC has multiple processor cores. A microcontroller, microprocessor, digital signal processor, or application specific instruction set processor can all be used.
- Secondly, for performing the computation the chip must have its memory. It could have memory in the form of RAM, ROM, EEPROM, or even flash memory.

- External interfaces are the next requirement, as they will allow it to conform to industry standard communication protocols such as USB, Ethernet, and HDMI. It can also make use of wireless technology and protocols such as WiFi and Bluetooth.
- For visualizing the interfaces, it must have a Graphical Processing Unit (GPU).
- Voltage regulators, phase lock loop control systems and oscillators, clocks and timers, analog to digital and digital to analog converters must also be included in SoC.
- For connecting all the individual blocks it must have an internal interface bus or a network.

1.2.1.4 Advantages of SoC

- Low power
- Low cost
- High reliability
- Small form factor
- High integration levels
- Fast operation
- Greater design
- Small size
- Low latency
- Better efficiency and performance
- Less time to market

1.2.1.5 Disadvantages of SoC

- More verification.
- Fabrication cost.
- Increased complexity.
- Time to market demands

1.2.1.6 SoC varieties

- **NVIDIA Tegra 3 is a graphics processor from NVIDIA**

  The NVIDIA Tegra 3 is a Tegra family SoC that is found in a variety of Android handsets. The Tegra 3 is used in some devices such as the Asus Eee Pad, HTC One X, and Google Nexus Tablet. This model includes a CPU with five cores. Each core is an ARM Cortex A9 chip, with the fifth core running at 500MHz and using a low-power silicon process.

- **Qualcomm's Snapdragon S4 processor**

  When it comes to Android smartphones and tablets, Qualcomm is crucial. It is powered by a processor that is similar to the ARM Cortex A15.

- **Samsung Exynos 4 Quad**

  The ARM architecture is used in this SoC. It has a quad-core ARM Cortex-A9 CPU and a 1.4GHz ARM Mali-400 MP4 quad-core GPU. This processor can handle a wide range of tasks, including 3D gaming, multitasking, and video recording and playback.

- **Intel Medfield**

  The Medfield SoCs from Intel are not based on the ARM architecture. These SoCs are built using x86 technology. Medfield SoCs can provide OEMs with a single-core processor running at 1.6-2GHz with a PowerVR SGX540 GPU.

- **OMAP 4 from Texas Instruments**

  The ARM Cortex A9 45nm architecture is used in the fourth generation of OMAPs. Motorola Atrix 2, Motorola Droid RAZR, LG Optimus 3D, and LG Optimus Max are some Android devices that employ this SoC.

1.2.1.7 SoC design challenges

The different SoC design challenges are given below:

1. Strategy for architecture
2. Strategy for test design
3. Strategy for validation
4. Backend Strategy and Synthesis
5. Integration Strategy
6. On chip Isolation

- **Strategy for Architecture**

  The type of processor we employ to create the SoC is an extremely significant issue to consider. Furthermore, the type of bus that must be used is a question of decision.

- **Strategy for Test Design**

  The majority of frequent physical problems are represented as faults in this approach. The essential circuitry incorporated in the SoC architecture assist in defect detection.

- **Strategy for validation**

  There are two primary concerns to consider while validating SoC designs. The first issue is that we need to double-check

the IP cores. The second issue is that we need to double-check the system's integration.

- **Backend Strategy and Synthesis**

  Many physical effects must be taken into account while planning the SoC synthesis and strategy. IR drop, cross talk, 3D noise, antenna effects, and EMI effects are all examples of effects. Chip planning, power planning, DFT planning, clock planning, timing, and area budgeting are all required early in the design process to address these difficulties.

- **Strategy for Integration**

  To create a smooth integration strategy, all of the above-mentioned data must be examined and combined.

- **On chip Isolation**

  Many factors must be considered in on-chip isolation, including the impact of process technology, grounding effects, guard rings, shielding, and on-chip decoupling.

1.2.1.8 SoC applications

Following are few applications of SoC

- **Market for mobile phones**

  The mobile industry, particularly in smartphones, is the most popular and basic application of SoC. Because smartphones are becoming slimmer and lighter as technology advances, the use of a SoC (whose size is shrinking at an alarming rate with each new generation) is the greatest fit for this change. Furthermore, high performance and low power consumption are two significant elements that influence smartphone performance, and SoC excels at both. As illustrated in this image, the A6 CPU was the first system on chip used in the deconstruction of the iPhone 5.

- **Embedded systems**

In the modern world, almost every microcontroller and CPU has a SoC operating on top of it. Component coupling is tighter, resulting in greater reliability and performance.

Embedded systems can be seen in Apple's smart watch. The apple S1 SoC is used in this smart watch.

The SoC in the Samsung Galaxy Gear is based on the ARM Cortex M4 microcontroller. For example STM32F401B.

- **Personal computers**

Another important application of the SoC is personal computers; many modern personal computers do not have a motherboard, instead relying on the SoC to provide great performance and minimal size.

1.2.1.9 Examples of SoCs

The majority of SoCs on the market today are ARM-based. Qualcomm's Snapdragon SoCs, Apple's A4, and Nvidia's Tegra series are some examples of smartphone SoCs. The Raspberry Pi 2 uses the Broadcom BCM2836 SoC. The Open Cores community has created a number of SoCs.

1.2.2 Structure of System on Chip: Design Flow

SoC design flow structure is as shown in the following figure 1.2

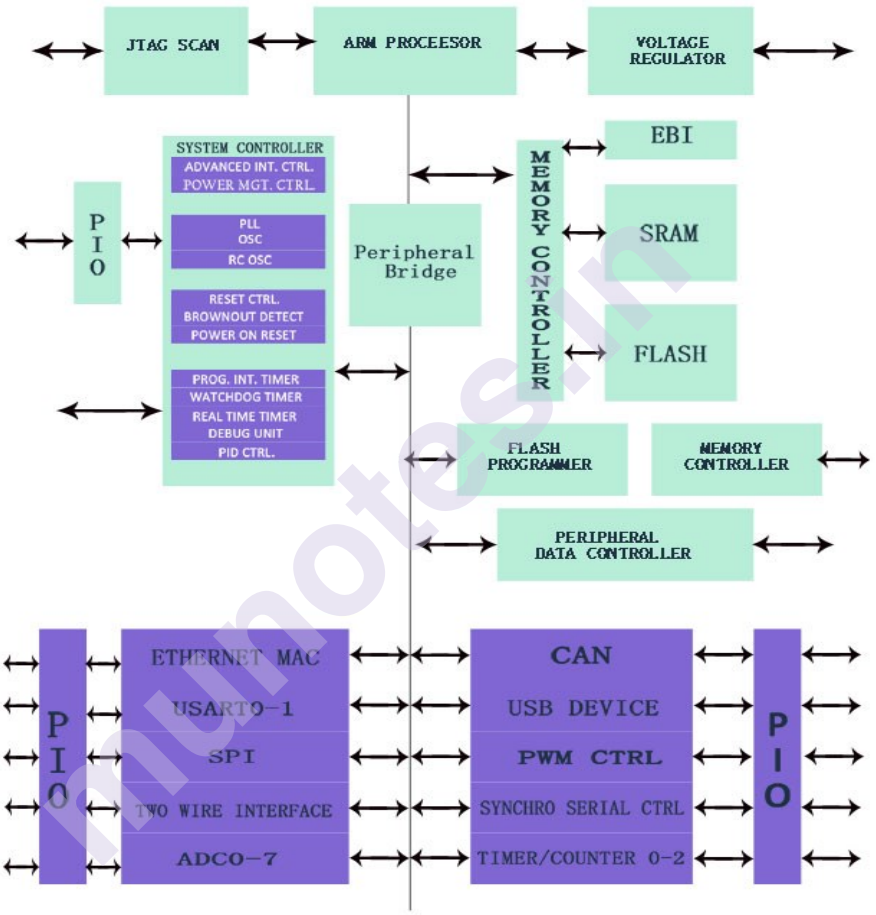


Figure 1.2 SoC design flow structure

The goal of the SoC design flow is to build the hardware and software of SoC designs. In general, the design pipeline of SoCs includes the following steps:

- Hardware and Software Modules: SoC hardware blocks are made up of pre-qualified hardware components and software modules that are combined in a software development environment. For the development of the modules, hardware description languages like as Verilog, VHDL, and SystemC are utilised.

- Functional Verification: Before being sent to the foundry, the SoCs are tested for logic accuracy.

- Verify hardware and software designs: Engineers have used FPGA, simulation acceleration, emulation, and other technologies to verify and debug the hardware and software of SoC designs.

- Place and Route: After the SoC has been debugged, the next step is to place and route the whole design onto the integrated circuit before it is sent to manufacture. Full custom, standard cell, and FPGA technologies are widely used in the fabrication process.

1.3 SOC PRODUCTS

This section describes various SoC products such as FPGA, GPU, APU and compute unit in detail.

1.3.1 FPGA

Field Programmable Gate Array is the abbreviation for FPGA. It's an integrated circuit that may be configured by a user after it's been created for a specific purpose. Adaptive logic modules (ALMs) and logic elements (LEs) are coupled via programmable interconnects in modern FPGAs. These blocks combine to form a physical array of logic gates that can be configured to execute a variety of tasks. This distinguishes them from other types of microcontrollers or Central Processing Units (CPUs), whose configuration is fixed and cannot be changed by the manufacturer. FPGA overview is shown in the figure 1.3 below

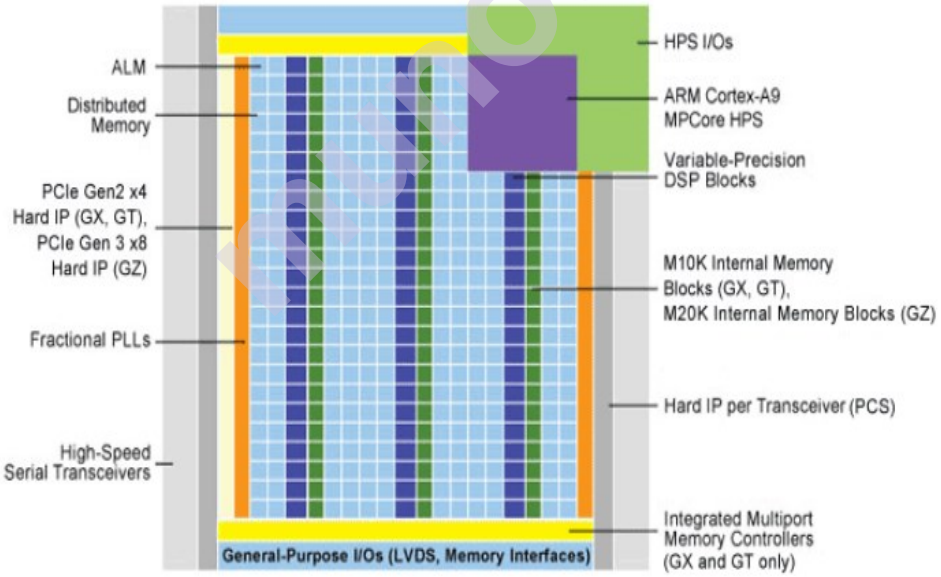


Figure 1.3 An overview of FPGA

The early programmable circuits were quite basic, consisting solely of logic gates. This was sufficient to execute a variety of logical functions with zeros and ones as inputs and outputs. Programmable circuits became increasingly and more powerful over time. You programme logic cells that can act as registers, adders, multiplexers, or lookup tables in programmable circuits.

While the circuit is running, the way the cells work and the structure of the cells can both be altered. A circuit can be reprogrammed to fulfil several roles, including those of an ARM processor, a network interface card, or a video encoder, to mention a few. Figure 1.4 shows the adaptive logic module of FPGA.

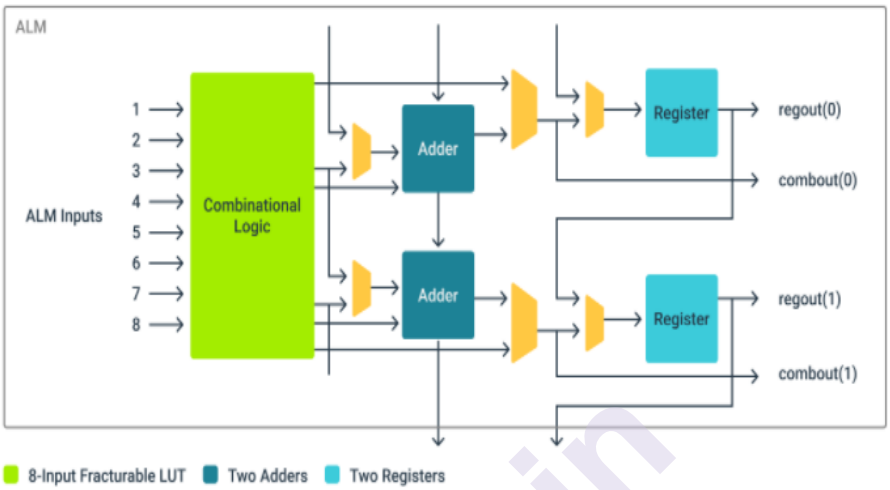


Figure 1.4 **Adaptive Logic Module of an Altera/Intel FPGA**

1.3.1.1 Working of FPGA

The FPGAs are made up of logical modules that are linked together through routing channels. Each module is made up of a programmable lookup table that is used to manipulate the elements that make up each cell and perform logical functions on those elements. Each cell, in addition to the lookup table, contains cascaded adders that allow addition to be performed. Subtraction can also be accomplished by altering the input's logical states. There are additionally registers (logical elements used to conduct the most basic memory functions) and multiplexers in addition to these (switching elements).

Depending on the manufacturer model, FPGAs can also incorporate static and dynamic on-chip memory. CPU cores, memory controllers, USB controllers, and network cards are among the ready-to-use components found in FPGAs. There is no need to include these components in the FPGA framework because they are so widely used. Instead, you can use a component that has already been made.

1.3.1.2 What can you do using FPGA programming?

FPGAs are primarily employed in the development of **application-specific integrated circuits** (ASICs). To begin, you must create the circuit's architecture. The prototype is then built and tested using an FPGA. Errors are reversible. Once the prototype has proven to be functional, an ASIC project based on the FPGA design is produced and fabricated. Because manufacturing an integrated circuit is a complex and time-consuming procedure, this helps you to save time. It also saves money because a single FPGA can handle multiple versions of the same project. In this regard, it's

worth noting that modern **Tensor Processing Units** (TPUs), often known as crypto currency miners, were first designed as FPGAs and only then built.

In real-time systems, when response time is critical, FPGAs are also used. Response time is not fixed in ordinary CPUs, thus you never know when you'll get a response once the initial signal comes. Real-time operating systems are used to reduce it or keep it within a certain range. Even yet, in cases requiring a quick response time (sub milliseconds), this falls short. To solve this problem, the requested algorithm must be implemented in FPGA using combinational or sequential logic to guarantee a consistent response time of less than milliseconds. Once ready, a real-time system designed in FPGA can be changed and pushed into production. This method will result in a considerably speedier and more energy-efficient integrated circuit.

FPGAs are also employed in applications where the hardware configuration is subject to change and a circuit that can adapt to these changes is required. FPGA becomes an obvious alternative if your hardware supplier's move and the new hardware does not have the needed interface.

1.3.1.3 How to program FPGAs?

It's possible that the term "FPGA programming" is a misleading. After all, unlike CPUs and GPUs, there is no true program to run sequentially. FPGA programming is designing a hardware architecture that can run a given algorithm and describing it using a hardware description language (HDL). As a result, unlike normal programs executed by CPUs or GPUs, the building blocks of this algorithm will not be a memory register and a set of operations to be done. Low-level elements such as logic gates, adders, registers, and multiplexers will make up a "FPGA program."

This provides you with a lot of versatility. If your data type is 20 bits, for example, you can only use 20-bit instructions to conduct operations. In the realm of CPUs, there are only manufacturer-set registers and instructions that cannot be modified. You can change to the data type in FPGAs, on the other hand, because you design the hardware architecture yourself.

You can also use general-purpose CPUs to perform processes that are either complex or time-consuming. CPUs, for example, conduct block cyphers and cryptographic tasks in many cycles, taking substantially longer than FPGAs.

1.3.1.4 Languages used in FPGA programming

Specific languages, like as VHDL or Verilog, are used to programme FPGAs. The syntax of VHDL is more close to Pascal than C, allowing for programming that is distinct from that of traditional high-level languages. Verilog, on the other hand, is similar to C, making it more intuitive and user-friendly for those with no prior familiarity with low-level programming.

VHDL is an obsolete language with a number of drawbacks, one of which being the difficulty in determining whether the architecture works as planned. Python is used to generate chunks of the code in various applications to make our lives easier. Of course, everything could be written in VHDL, but Python is more user-friendly.

The HDL simulator is the most important tool for designing hardware. It enables you to simulate how the architecture functions using sample input data. As a result, you can see how the data flows. The HDL simulator is particularly important since compiling a given hardware description into an FPGA board and programming the board itself, even for a basic programme, might take a long time. You can use the simulator to extensively test the algorithm you wish to put on an FPGA board.

1.3.1.5 FPGA Architecture

A typical FPGA design (Figure 1.5) is made up of thousands of basic elements called configurable logic blocks (CLBs), which are connected by a system of programmable interconnects called a fabric that routes signals between CLBs. The FPGA and external devices are connected using input/output (I/O) blocks.

The CLB is also known as a logic block (LB), a logic element (LE), or a logic cell (LC), depending on the manufacturer.

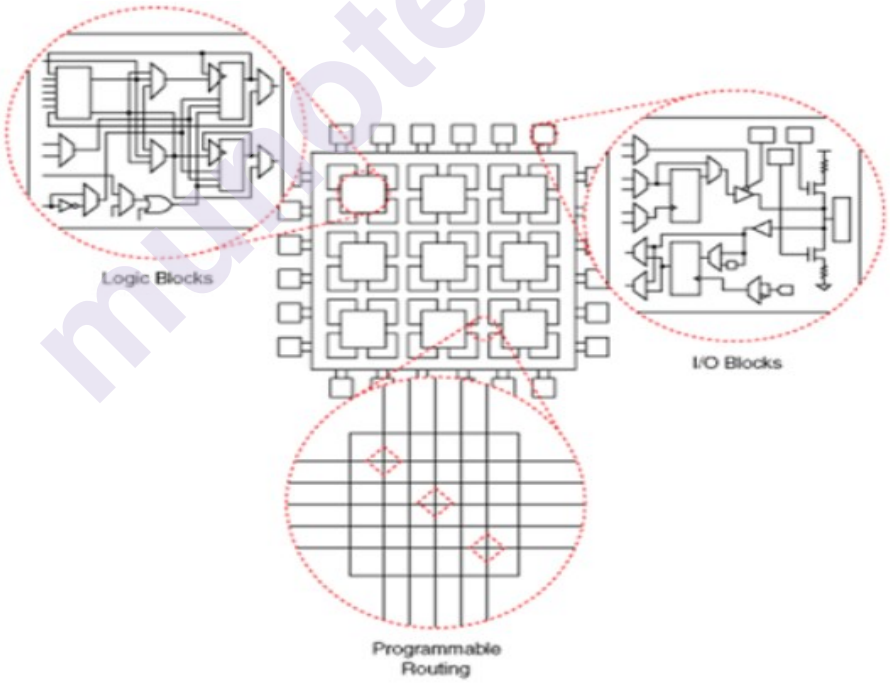


Figure 1.5: The fundamental FPGA architecture

A CLB is made up of numerous logic blocks (see Figure 1.6). An FPGA's lookup table (LUT) is a distinguishing feature. LUTs with four to six input bits are commonly utilised, and they hold a predefined list of logic outputs for any combination of inputs. Multiplexers (mux), full adders (FAs), and flip-flops are all commonly used logic functions.

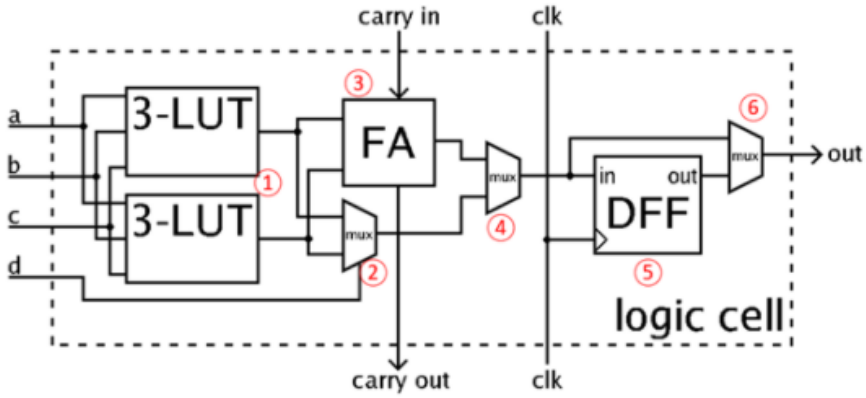


Figure 1.6: *A simplified CLB: The four-input LUT is formed from two three-input units*

The number and location of components in the CLB vary depending on the device; in the simplified example in Figure 1.6, during FPGA programming, two three-input LUTs (1), an FA (3), and a D-type flip-flop (5), as well as a standard mux (2) and two muxes, (4) and (6), are configured.

There are two modes of operation for this reduced CLB. The LUTs are merged with Mux 2 to produce a four-input LUT in normal mode, and the LUT outputs are provided as inputs to the FA along with a carry input from another CLB in arithmetic mode. Mux 4 switches between the FA and LUT outputs. The D flip-flop in Mux 6 decides whether the operation is asynchronous or synchronised to the FPGA clock.

CLBs in current-generation FPGAs can combine for more complicated operations such as multipliers, registers, counters, and even digital signal processing (DSP) capabilities; CLBs can combine for more sophisticated operations such as multipliers, registers, counters, and even DSP functions.

1.3.1.6 FPGA Applications

FPGAs are well-suited to a variety of markets because to their programmability. Xilinx, as the industry leader, provides wide range of solutions that include FPGA devices, powerful software, and configurable, ready-to-use IP cores for markets and applications including:

- Aerospace and Defense - FPGAs for image processing, waveform synthesis, and partial reconfiguration for SDRs that are radiation-tolerant.
- ASIC Prototyping - ASIC prototyping with FPGAs allows for quick and accurate SoC system modelling and embedded software verification.

- Automobiles - Silicon and IP solutions for gateway and driving assistance systems, as well as comfort, convenience, and in-vehicle infotainment.
- Broadcast & Pro AV - With Broadcast Targeted Design Platforms and solutions for high-end professional broadcast systems, you can adapt to changing requirements faster and extend product life cycles.
- Consumer Electronics - Affordably priced technologies that enable next-generation, full-featured consumer applications such convergent handsets, digital flat panel display, information appliances, home networking, and residential set-top boxes.
- Data Center - Designed for high-bandwidth, low-latency servers, networking, and storage applications to boost cloud deployment value.
- Network Attached Storage (NAS), Storage Area Network (SAN), servers, and storage appliances solutions for high-performance computing and data storage.
- Industrial - Xilinx FPGAs and targeted design platforms for Industrial, Scientific, and Medical (ISM) enable higher degrees of flexibility, faster time-to-market, and lower overall non-recurring engineering costs for a wide range of applications including industrial imaging and surveillance, industrial automation, and medical imaging equipment (NRE).
- Wired Communications - Wired Communications' end-to-end solutions cover reprogrammable networking linecard packet processing, framer/MAC, serial backplanes, and more.
- Wireless Communications - RF, baseband, connection, transport, and networking solutions for wireless equipment, including WCDMA, HSDPA, WiMAX, and other standards.

1.3.1.7 Artificial Intelligence: The next frontier for FPGAs

FPGAs are now gaining traction in another field: artificial intelligence (AI) using deep neural networks (DNNs) (AI). It needs a lot of processing resources to run DNN inference models. GPUs are frequently used to speed up inference processing, but in some circumstances, high-performance FPGAs may surpass GPUs when it comes to evaluating massive amounts of data for machine learning.

Microsoft is already utilizing Intel FPGA flexibility for AI acceleration. Customers can use Microsoft Azure cloud services to access Intel Stratix FPGAs as part of Project Brainwave. These FPGAs have been configured specifically for running deep learning models on cloud servers with these FPGAs. Developers can use the Microsoft service to tap into the capabilities of FPGA chips without having to buy or configure additional gear or

software. Developers can instead use open-source tools like the Microsoft Cognitive Toolkit or the Tensor Flow AI programming framework.

1.3.1.8 Benefits by using FPGAs

i] Flexibility

- Every time the device is powered up, the FPGA functionality can vary. So, if a design engineer wants to make a modification, all they have to do is download a new configuration file into the device and try it out.
- Frequently, updates to the FPGA can be made without the need for costly PC board replacements.
- ASSPs and ASICs have fixed hardware functionality that cannot be modified without a significant financial and time investment.

ii] Acceleration

- Improve your system's performance and/or get items to market faster.
- In comparison to ASICs, FPGAs are available “off the shelf” (which require manufacturing cycles taking many months).
- OEMs can deploy systems as soon as the design is operational and proven because to FPGA flexibility.
- FPGAs provide CPUs with off-load and acceleration features, allowing the entire system to run faster.

iii] Integration

- On-die CPUs, transceiver I/Os at 28 Gbps (or faster), RAM blocks, DSP engines, and other features are available in today's FPGAs. More functionality in the FPGA mean fewer devices on the circuit board, which improves reliability by decreasing device failures.

iv] Total Cost of Ownership (TCO)

- While ASICs are less expensive per unit than FPGAs, they require a non-recurring expense (NRE), expensive software tools, specialist design teams, and extended production cycles to produce.
- Long lifecycles (15 years or more) are supported by Intel FPGAs, eliminating the cost of rebuilding and requalifying OEM production equipment whenever one of the electronic components on-board becomes obsolete (EOL).

- FPGAs lower risk by allowing prototype systems to be shipped to clients for field testing while still allowing for quick changes before ramping up to volume production.

1.3.2 GPU

The CPU (central processing unit) is referred to as a computer's brain and GPU its soul. GPUs, on the other hand, have broken out from the limits of the PC over the last decade.

GPUs have sparked a global AI craze. They've evolved into an important component of current supercomputing. They've been incorporated into new hyper scale data centers that are expansive. They've evolved into accelerators, speeding up everything from cryptography to networking to artificial intelligence.

They also continue to drive gaming and professional graphics advancements in workstations, desktop PCs, and a new generation of laptops.

1.3.2.1 What is a GPU?

GPUs (graphics processing units) are now much more than the PCs in which they first appeared, but they are still based on a much older concept known as parallel computing. GPUs are extremely powerful because of this.

CPUs, to be sure, are still necessary. CPUs are quick and versatile, and they race through a series of tasks that require a lot of interaction. For example, retrieving data from a hard drive in response to a user's keystrokes.

GPUs, on the other hand, divide complex problems into thousands or millions of smaller tasks and solve them all at once. This makes them ideal for graphics, where textures, lighting, and shape rendering must all be done at the same time to keep images moving across the screen.

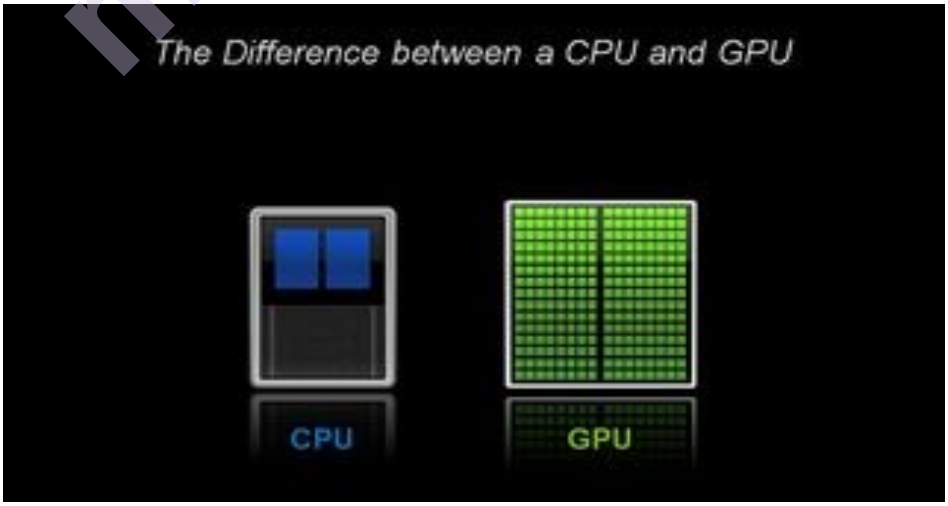


Figure 1.7: Difference between CPU and GPU

1.3.2.2 Difference between CPU and GPU

CPU	GPU
Full Form: Central Processing Unit	Full Form: Graphics Processing Unit
Few cores	Many cores
Low latency	High throughput
Good for serial processing	Good for parallel processing
Can do a limited operations at once	Can do thousands of operations at once

Because the CPU only has a few cores and a lot of cache memory, it can only process a few software threads at a time, whereas the GPU consisting of hundreds of cores thus handling thousands threads at once.

Parallel computing, which was once an esoteric technology, is now available thanks to GPUs. It's a technology with an illustrious pedigree that includes names like Seymor Cray, the father of supercomputing. GPUs put this idea to work in the desktops and gaming consoles of over a billion gamers, rather than taking the form of hulking supercomputers.

1.3.2.3 What does a GPU do?

The graphics processing unit (GPU) has emerged as one of the most important types of computing technology for both personal and business computing. The GPU, which was created for parallel processing, is used in a variety of applications, including graphics and video rendering. GPUs are becoming more popular for use in creative production and AI, despite being best known for their gaming capabilities.

GPUs were created with the intention of speeding up the rendering of 3D graphics. They improved their capabilities by becoming more flexible and programmable over time. With advanced lighting and shadowing techniques, graphics programmers were able to create more interesting visual effects and realistic scenes. Others began to use GPUs to dramatically speed up additional workloads in high-performance computing (HPC), deep learning, and other areas.

1.3.2.4 Unified GPU Architecture

A parallel array of multiple programmable processors encourages unified GPU architectures. Unlike earlier GPUs, which had distinct processors specialized to each processing type, they combine vertex, geometry, and pixel shader processing and parallel computing on the same processors. For texture filtering, rasterization, raster operations, anti-aliasing, compression, decompression, display, video decoding, and high-definition video

processing, the programmable processor array is intimately linked with fixed function processors.

Many core GPUs have a distinct architectural design point than multicore CPUs, focusing on efficiently running many parallel threads on many processor cores. More of the per-chip transistor budget is given to computation, and less to on-chip caches and overhead, by using many simpler cores and optimizing for data-parallel behavior among groups of threads.

The logical pipeline, which consists of discrete independent programmable stages, is mapped onto a physical dispersed array of processors in shown in the figure 1.8.

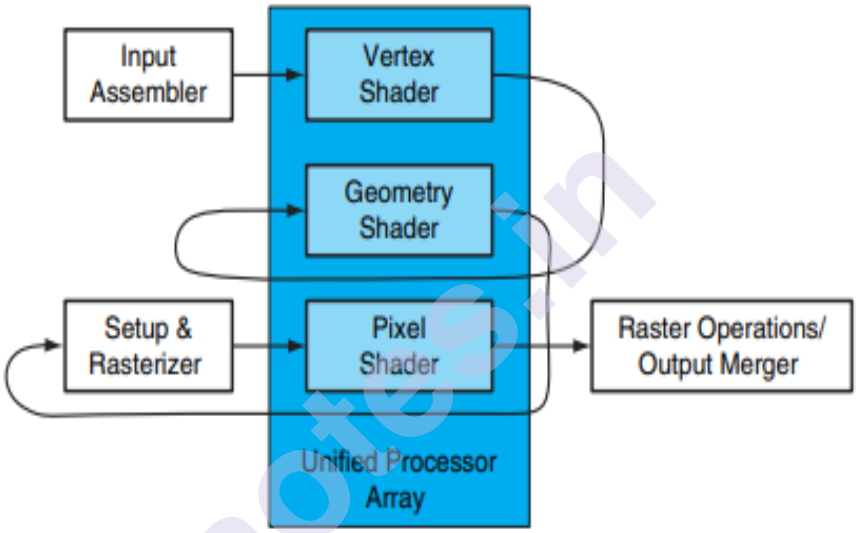


Figure 1.8 Logical pipeline mapped to physical processors. On the array of unified processors, the programmable shader stages run, and the logical graphics pipeline dataflow recirculates via the processors.

Processor Array

Many processor cores are grouped into multithreaded multiprocessors in a unified GPU processor array. A GPU with 112 streaming processor (SP) cores structured as 14 multithreaded streaming multiprocessors (SM) is shown in Figure 1.9. Each SP core is extremely multithreaded, with 96 concurrent threads and their hardware states to manage. An interconnection network connects the CPUs to four 64-bit-wide DRAM partitions. Each SM is equipped with eight SP cores, two SFUs, instruction and constant caches, a multithreaded instruction unit, and shared memory. This is the NVIDIA GeForce 8800's implementation of the Tesla architecture. Traditional graphics applications such as vertex, geometry, and pixel shading run on the unified SMs and SP cores, while computation programs operate on the same processors.

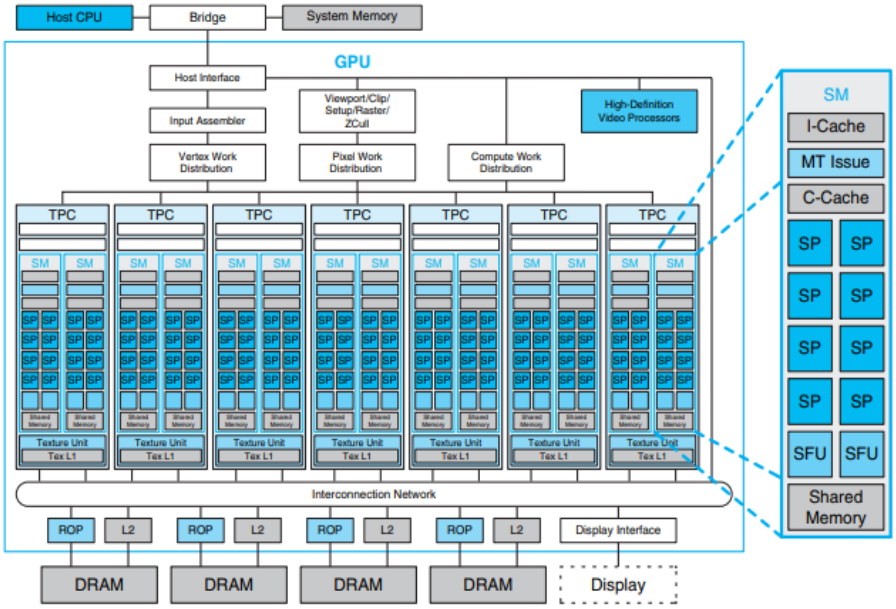


Figure 1.9 Basic unified GPU architecture

By adjusting the number of multiprocessors and memory partitions, the processor array design can be scaled to smaller and bigger GPU systems. Seven clusters of two SMs share a texture unit and a texture L1 cache in Figure 1.9. Given a set of coordinates into a texture map, the texture unit returns filtered results to the SM. Because the filter regions of support for subsequent texture requests frequently overlap, a tiny streaming L1 texture cache can help minimize the number of queries to the memory system. A GPU-wide interconnection network connects the processor array to raster operation (ROP) processors, L2 texture caches, external DRAM storage, and system memory. The number of processors and memories in a GPU system can be scaled to meet the needs of various performance and market sectors.

1.3.2.5 Applications of GPU

GPUs were largely employed to accelerate real-time 3D graphics applications, such as gaming, two decades ago. Computer experts believed that GPUs had the potential to address some of the world's most complex computing challenges as the twenty-first century began.

The general-purpose GPU era began as a result of this insight. Graphics technology is now being used to solve an ever-widening range of challenges. GPUs are more programmable than they've ever been, allowing them to speed up a wide range of applications beyond graphics rendering.

- GPUs for gaming: With hyper realistic graphics and enormous, sophisticated in-game worlds, video games have become more computationally intensive. With the rise of virtual reality games and improved display technologies such as 4K panels and high refresh rates, graphics processing demands are rapidly increasing. GPUs can render visuals in both 2D and 3D modes. Games can be played at

greater resolutions, quicker frame rates, or both with superior visual performance.

- GPUs for Video Editing and Content Creation: Long rendering times have plagued video editors, graphic designers, and other creative professions for years, sapping computing resources and stifling creative flow. GPUs' parallel processing now makes rendering video and graphics in higher-definition formats faster and easier.

- GPUs for Machine Learning: AI and machine learning are two of the most interesting applications for GPU technology. GPUs can give amazing acceleration in workloads that take use of GPUs' highly parallel nature, such as image recognition, because they have such a large amount of processing capability. Many of today's deep learning solutions rely on GPUs and CPUs working together.

1.3.3 APU

An APU is a 64-bit microprocessor that combines the processing capabilities of a CPU (Central Processing Unit) and a GPU (Graphics Processing Unit) onto a single chip. While APU may sound like any other computer processor, it is only used by AMD as the brand name for the CPU/GPU combo chips they produce. To understand what an APU is, it's helpful to know a little about the two CPUs it combines.

The CPU, also known as the "brain" of the computer, is the main processing unit that receives and executes instructions from software or applications. It also transmits instructions to other elements of the system, instructing them on what they should do. It is the most important component of a computer system; without it, the computer would be rendered useless.

The GPU performs comparable tasks to the CPU, but it only handles graphics-related data and generates graphical output. A computer without a GPU is blind, with no video output, just as a computer without a CPU is.

The CPU and GPU are two independent components in most systems. Except that the data transfer rate will improve if the two processors are closer to each other, there isn't much of a problem with this. Furthermore, having these two units running at the same time results in higher power usage, which AMD is well aware of. They released their first high-performance and energy-efficient processor, the APU, in 2011, which merged the benefits of the CPU and GPU into a single chip.

1.3.3.1 Evolution of APU

AMD has been developing structured and efficient architecture for their CPUs and GPUs as a major manufacturer of computer hardware. Their APUs is usually a combination of their existing CPU and GPU designs. The resulting processor outperforms the typical CPU and GPU when used together. It was formerly known as the "Fusion" before being renamed the "APU." The term was eventually changed to APU because to a trademark infringement concern.

AMD makes two kinds of APUs: one for high-performance devices and the other for low-power devices. Llano was the codename for the first generation APU for high-performance devices, which contained K10 CPU cores and a Radeon HD 6000-series GPU. Similarly, the first APU for low-power devices, codenamed Brazos, had the Bobcat microarchitecture and a Radeon HD 6000-series GPU. Trinity, AMD's second generation of high-performance APU, and Brazos 2.0, AMD's second generation of low-power APU, were released in 2012. As AMD's CPU and GPU architecture improved, the APU improved as well, with performance at the forefront of each improvement. Following versions used the most up-to-date architecture available at the time, and each iteration saw significant advancements over the previous one. Apart from performance, AMD has increased the upgradability of its products. Previously, future CPU upgrades were not conceivable, but starting with the APU Ryzen series, this was no longer the case. Renoir, the 2020 release, is built on the Zen 2 core architecture and features Vega 8 graphics.

APUs are still evolving today, and with AMD's newest and more powerful architectures, the next generation of APUs is on the way.

1.3.3.2 Benefits over CPU and GPU

The game-changing technology of the APU is a key advancement in the computing industry, with various advantages over the CPU + GPU configuration.

Improved performance: The data transfer rate was greatly enhanced by combining the CPU and GPU in the same chip because they now share the same bus and resources. OpenCL (Open Computer Language), a standard interface for parallel computing that makes use of the computing power supplied by GPUs, is also supported by APUs. Tasks that demand the high processing power of a CPU and the fast image processing of a GPU can benefit from the performance of an APU's multi-core CPU and GPU.

Power-efficient: Not only do combining two chips reduce space, but it also saves electricity. Apart from enhancing the APU's performance, AMD is constantly working to reduce the chip's power consumption, despite the fact that it is already low. Low Thermal Design Power is a feature of newer models (TDP). The Ryzen Embedded 1102G, for example, has the lowest TDP of only 6W.

Cost-effective: The cost benefit of AMD's APU over a CPU and GPU combination is arguably the most significant. Buying an APU is often less expensive than buying a CPU and GPU individually, with prices ranging from $100 to $400 depending on the specifications. Though the higher-end components are more expensive, they are still less expensive than the cost of a CPU and GPU with the same level of performance. This applies to future upgrades as well. Due to AMD's permissive attitude toward APU upgradability and compatibility, consumers can save a lot of money by replacing just one processor rather than both.

1.3.3.3 What sectors can benefit from APUs?

Accelerated Processing Units have been used in a variety of areas, including:

- **Software Development**

  Software developers can employ APUs to create heterogeneous computing architectures that blend CPU and GPU technology. This combination allows them to work on projects that require a high level of speed and processing power. Today's APUs also supports Open Computing Language (OpenCL) pictures, which helps. OpenCL is a standard interface for task- and data-based parallelism in parallel computing. The majority of activities necessitate a lot of computer power (from CPUs) and quick picture processing (a GPU feature). However, CPUs and GPUs rarely process data at the same time. The process is sped up by APUs, which combine both capabilities and allow parallel processing.

  APUs are also less expensive than buying a CPU and GPU, making them perfect for software developers who don't need a lot of processing power.

- **Visual content creation**

  The majority of today's digital material is mainly visual. Digital content creators may quickly create high-quality videos that elevate the user experience with an APU-powered computer.

  Advanced Micro Devices (AMD), the company that invented APUs, allows content makers to employ built-in universal video decoders (UVDs) to enhance video content so that it may be displayed on a big screen without losing quality.

  APUs allow content creators to clean up photos and movies in addition to offering high-quality displays, simplifying and streamlining the content creation process.

- **Gaming**

  APUs for gaming are also handy for gamers who want to build their own computers. These enable them to take advantage of improved and quicker graphics processing, enhancing their gaming experience without breaking the bank.

1.3.3.4 Is it a Better Processor?

APUs have been found in a variety of devices, including desktops, laptops, servers, mobile phones, and gaming consoles. For a decade, businesses and consumers have supported this heterogeneous chip. Can it, however, truly replace the CPU and GPU? In the end, it would be determined by the wants and demands of the user.

Consumers, PC builders, and budget gamers can take use of APU's advantages. The majority of APUs are capable of delivering adequate performance. In fact, it has the ability to exceed mid-range CPUs and GPUs. It's an excellent alternative for customers who don't want intense graphics or the maximum available CPU speed. It will also work well with ordinary PCs at home and in the business. AMD continues to create sophisticated APUs, with latest models capable of handling graphics-intensive tasks.

When it comes to intensive gaming, though, an APU will not suffice. It's still unable to match the graphical experience provided by high-end discrete graphics cards. An APU, on the other hand, would be an excellent choice for low-budget, entry-level PC building and gaming.

Although an APU cannot totally replace the CPU and GPU, it is a suitable high-performance, low-power option in many circumstances. As AMD's designs improve and new technologies emerge, it wouldn't be surprising if future generations of the APU can completely replace both the CPU and the GPU.

1.3.4 Compute Units

Compute units are comparable to host groups, but they have the added feature of granularity, allowing cluster-wide structures that mimic network architecture to be constructed. Task scheduling that considers processing unit resource needs optimizes job placement based on the underlying system architecture, eliminating communications bottlenecks. When conducting communication-intensive parallel operations across multiple hosts, compute units are extremely handy. Compute units represent the topology of a cluster network for workloads that require a lot of communication between processes. Computing units, for example, can help reduce network latency and take use of fast interconnects by putting all job operations in the same rack, rather than making several network hops.

Availability of resources Strings can be used to indicate compute unit requirements such as performing a job solely (excl), evenly distributing a job across many compute units (balancing), or selecting compute units depending on other criteria.

A computation unit is made up of 64 shader processors and four TMUs. The compute unit is independent from the render output units, yet it feeds into them (ROPs). A CU Scheduler, a Branch & Message Unit, four SIMD Vector Units (each 16-lane wide), four 64KiB VGPR files, one scalar unit, a 4 KiB GPR file, a 64 KiB local data share, four Texture Filter Units, sixteen Texture Fetch Load/Store Units, and a 16 KiB L1 Cache are all contained in each Compute Unit. A 16KB L1 instruction cache and a 32KB L1 data cache are shared by four computing units and are both read-only. A SIMD-VU can process 16 items at a time (per cycle), but an SU can only process one element at a time (per cycle). In addition, the SU performs other tasks like as branching.

Every SIMD-VU has its own private memory where its registers are stored. There are two sorts of registers: scalar registers (s0, s1, etc.), which contain

four bytes of data, and vector registers (v0, v1, etc.), which hold 64 bytes of data. Every operation on the 64 numbers in the vector registers is performed in simultaneously. When you work with them, you're truly working with 64 inputs. For instance, suppose you're working on 64 separate pixels at the same time (for each of them the inputs is slightly different, and thus you get slightly different color at the end). There are 512 scalar registers and 256 vector registers in each SIMD-VU.

1.3.4.1 Compute unit configuration

Compute unit configuration must meet the following requirements to ensure consistency:

- Hosts and host groups are only found in the highest granularity compute unit type.
- At most one compute unit of the finest granularity's membership list contains hosts.
- The same type of compute units (or hosts) is members of all compute units of the same type.

1.3.4.2 Where to use compute units?

The following parameters in LSF configuration files can be defined using LSF compute units:

- The compute unit type allowed for the queue is EXCLUSIVE in lsb.queues.
- The hosts on which jobs from this queue can be run are listed in lsb.queues as HOSTS.
- RES REQ in lsb.queues is used to track resource requirements for queue compute units.
- For application profile compute unit resource needs, see RES REQ in lsb.applications.

1.3.4.3 Different configurations of compute unit

Customers can select from the following compute unit configurations based on their requirements as shown in the following Table I:

Table I: Compute unit configurations

Compute Unit	Configuration	Size Parameter Value
Lite edition	1 CPU Core; 3072 MB Main memory	lite
Professional edition	2 CPU Cores; 4096 MB Main memory	pro

Compute Unit	Configuration	Size Parameter Value
Premium edition	4 CPU Cores; 8192 MB Main memory	prem
Premium Plus edition	8 CPU Cores; 16384 MB Main memory	prem-plus

1.4 ARM 8 ARCHITECTURE

1.4.1 SoC on ARM 8

A RISC processor is what the ARM processor is. Around 1980, the RISC was born out of processor development programs at Stanford and Berkeley universities. Between 1983 and 1985, Acorn Computers Limited in Cambridge, England, developed the ARM processor. It was the first commercially available RISC CPU, and it differs significantly from subsequent RISC architectures.

ARM Limited was founded in 1990 as a distinct company with the sole purpose of expanding the use of ARM technology. Since then, the ARM has been licensed to a number of semiconductor manufacturers throughout the world. It has established itself as an industry leader in low-power, low-cost embedded applications. Without the support of hardware and software development tools, no processor is very valuable. An instruction set emulator for hardware modeling and software testing and benchmarking, an assembler, C and C++ compilers, a linker, and a symbolic debugger are all part of the ARM toolset.

The Acorn RISC Machine

Between October 1983 and April 1985, Acorn Computers Limited in Cambridge, England, created the first ARM processor. ARM stood for Acorn RISC Machine at the time and until the foundation of Advanced RISC Machines Limited (later called simply ARM Limited) in 1990. Because of the popularity of the BBC (British Broadcasting Corporation) microcomputer, Acorn had established a strong position in the UK personal computer market. The BBC micro was an 8-bit microprocessor-based machine that quickly established itself as the dominant machine in UK schools following its launch in January 1982 in support of a series of BBC television programs. It also received enthusiastic support from the hobbyist community and was adopted by a number of research labs and higher education institutions.

Following the success of the BBC micro, Acorn's developers looked at different microprocessors to use in a successor computer, but all of the commercial options were missing. In 1983, 16-bit CISC microprocessors were available, although they were slower than ordinary memory

components. They also featured instructions that required many clock cycles (in some cases hundreds of clock cycles) to complete, resulting in extremely lengthy interrupt latency. The BBC micro profited immensely from the 6502's fast interrupt response, thus Acorn's designers were adamant that this feature of the processor's performance not be compromised.

The design of a proprietary microprocessor was contemplated as a result of these problems with commercial microprocessor products. The main stumbling issue was the fact that the Acorn team was well aware that commercial microprocessor programmes had consumed hundreds of man-years of design time. Because Acorn was a small company with only about 400 people, it couldn't consider such a large investment. It had to come up with a superior design in a fraction of the time, with no prior experience in bespoke chip design other than a few modest gate arrays for the BBC micro.

The papers on the Berkeley RISC I sprang out of nowhere in this seemingly improbable scenario. This was a processor that had been built in less than a year by a few postgraduate students and was competitive with the leading commercial products. There were no complex instructions to compromise the interrupt latency because it was fundamentally simple. It also came with supporting arguments that suggested it could be a harbinger of things to come, yet technical merit, no matter how strongly backed by academic reasoning, is no guarantee of commercial success.

The ARM was born as a result of a fortunate confluence of events, and it went on to become the main component of Acorn's product line. It later contributed its name to the firm founded to expand its market beyond Acorn's product line following a careful revision of the acronym expansion to Advanced RISC Machine. Despite the name change, the architecture is still quite similar to the Acorn design.

1.4.2 ARM 8 Architecture Introduction

1.4.2.1 Architectural inheritance

The Berkeley RISC I and II and the Stanford MIPS (which stands for Microprocessor without Interlocking Pipeline Stages) were the only examples of RISC architectures at the time the first ARM chip was designed, though some earlier machines, such as the Digital PDP-8, the Cray-1, and the IBM 801, which predated the RISC concept, shared many of the characteristics that later came to be associated with the RISC concept.

A number of Berkeley RISC design concepts were included into the ARM architecture, although others were discarded. A load-store architecture, fixed-length 32-bit instructions, and 3-address instruction formats were all used.

The following features were used on Berkeley RISC concepts that were rejected by ARM designers:

- **Register windows**

The Berkeley RISC processors' register banks contained a vast number of registers, with 32 of them visible at any given moment. The visible 'window' was shifted to provide each operation access to fresh registers, minimizing the data traffic between the CPU and memory caused by register saving and restoring.

The main issue with register windows is the high amount of chip space used up by the large number of registers. Although the shadow registers used to manage exceptions on the ARM are not too dissimilar in concept, this functionality was rejected on cost concerns.

Because it was included in the Berkeley prototypes in the early days of RISC, the register window technique was firmly connected with the RISC concept, although only the Sun SPARC architecture has embraced it in its original form since then.

- **Delayed branches**

Branches in pipelines obstruct the smooth flow of instructions, producing problems. Most RISC processors address the issue by employing delayed branches, which take effect after the next instruction has completed. Delay branches have the drawback of removing the atomicity of individual instructions. They function well on single-issue pipelined processors, but they don't scale well to super-scalar implementations and can cause problems when combined with branch prediction methods.

Delay branches were not utilized on the original ARM because they made exception handling more complicated; however, this has proved out to be a smart decision in the long run because it simplifies re-implementing the architecture with a new pipeline.

- **Single-cycle execution of all instructions**

Although the ARM can process most data in a single clock cycle, many other instructions require numerous clock cycles. The reasoning behind this was based on the fact that even a basic load or store instruction requires at least two memory accesses when using a single memory for both data and instructions (one for the instruction and one for the data). As a result, single-cycle operation of all instructions is only achievable with separate data and instruction memory, which were deemed too costly for the ARM application areas.

Instead of executing all instructions in a single cycle, the ARM was designed to use the fewest amount of cycles possible for memory accesses. Where this was more than one, the extra cycles were employed to do something beneficial, such as enable auto-indexing addressing modes, whenever possible. This minimizes the overall amount of ARM instructions needed to complete any given series of operations, resulting in improved performance and code density.

The need to keep the design basic was a major priority for the original ARM design team. Acorn designers had only worked with gate arrays with complexities of up to 2,000 gates prior to the first ARM processors; therefore the full-custom CMOS design medium was treated with caution. When travelling into unfamiliar area, it's best to limit the hazards that you can control, because there are still major risks from things that aren't well understood or fundamentally uncontrollable.

The ARM's simplicity is more visible in the hardware structure and implementation than in the instruction set architecture. From the perspective of the programmer, it manifests itself as conservatism in the ARM instruction set design, which, although adhering to the essential principles of the RISC approach, is less radical than many subsequent RISC designs.

The ARM's power-efficiency and tiny core size are due to the combination of basic hardware with an instruction set that is based on RISC ideas but preserves a few essential CISC elements, resulting in a substantially higher code density than a pure RISC.

1.4.2.2 About ARM Architecture

The ARM architecture, defines the behaviour of an abstract machine known as a Processing Element, or PE for short. Implementations that follow the ARM architecture must follow the Processing Element's defined behaviour. It is not intended to specify how to construct a PE implementation or to limit the scope of such implementations to the behaviours stated.

The programmer-visible behaviour of an implementation that is consistent with the ARM architecture must be the same as a simple sequential execution of the program on the processor element, unless the architecture specifies otherwise. The execution time of the program is not included in this programmer-visible behaviour.

- **An associated debug architecture**
- Corresponding trace architectures, which describe trace macrocells that implementers can implement with the associated processor hardware, are all defined in the ARM architecture.

The ARM architecture is RISC architecture with the following RISC architecture characteristics:

- A big file of uniform registers.
- A load/store architecture, in which data-processing operations are performed on register contents rather than memory contents directly.
- Modes with simple addressing, where all load/store addresses are determined only by register contents and instruction fields

The architecture specifies how the Processing Element interacts with memory, which includes caches, as well as a memory translation

mechanism. It also explains how several Processing Elements in a system interact with one another and with other observers.

The ARM architecture allows for implementations at a variety of performance levels. The ARM architecture is known for its small implementation size, high performance, and low power consumption.

Backwards compatibility, paired with the freedom to implement in a wide range of conventional and specific use cases, is a key characteristic of the ARMv8 architecture.

- AArch64, a 64-bit execution state compatible with prior versions of the ARM architecture
- AArch32, a 32-bit execution state compatible with previous generations of the ARM architecture

1.4.2.3 Architecture Profiles

Since its introduction, the ARM architecture has changed tremendously, and ARM continues to improve it. To date, eight major versions of the architecture have been defined, with version numbers ranging from 1 to 8. The first three versions are no longer in use.

The 64-bit and 32-bit execution states are referred to as AArch64 and AArch32, respectively.

- **AArch64:** This is the 64-bit execution state, which means that addresses are stored in 64-bit registers and that instructions in the base instruction set can operate 64-bit registers. The A64 instruction set is supported by the AArch64 state.
- **AArch32:** This is the 32-bit execution state, which means that addresses are stored in 32-bit registers and instructions in the base instruction sets are processed using 32-bit registers. The T32 and A32 instruction sets are supported by the AArch32 state.

Three architecture profiles are defined by ARM:

A: Application profile

- Supports a Memory Management Unit-based Virtual Memory System Architecture (VMSA) (MMU)
- AArchv8-A is the name given to an ARMv8-A implementation.
- The A64, A32, and T32 instruction sets are supported.

R: Real time profile

- Based on a Memory Protection Unit in real-time (MPU) it provides support for Protected Memory System Architecture (PMSA).
- Both A32 and T32 instruction sets are supported.

M: Microcontroller profile

- Provides a programmers' model for low-latency interrupt processing, including hardware register stacking and support for interrupt handlers written in high-level languages.
- Implements an R-profile PMSA variation.
- Supports a T32 instruction set variant.

1.4.2.4 ARMv8 architectural concepts

ARMv8 makes significant improvements to the ARM architecture while keeping a high level of compatibility with earlier versions.

The subsections that follow explain fundamental ARMv8 architectural concepts. Each section begins with an explanation of the concepts that are used to describe the architecture:

i] Execution state

- The PE execution environment is defined by the Execution state, which includes:
- The supported register widths.
- The instruction sets that are supported.
- Important features of:
  - The model of the outlier.
  - The Architecture of the Virtual Memory System (VMSA).
  - The model of the programmers.

The following are the execution states:

- **AArch64**
  - The state of 64-bit execution
  - Provides a 64-bit program counter (PC), stack pointer (SPs), and exception link registers
  - Provides 31 64-bit general-purpose registers, of which X30 is utilized as the procedure link register (ELRs)
  - Supports SIMD vector and scalar floating-point calculations with 32 128-bit registers
  - Has a single instruction set, A64
  - Support for 64-bit virtual addressing
  - Defines a number of PSTATE components that retain PE state
  - Defines the ARMv8 Exception model, with up to four Exception levels, EL0 - EL3, that give an execution privilege hierarchy Instructions that operate directly on certain PSTATE elements are included in the A64 instruction set

- Each system register is given a suffix that corresponds to the lowest Exception level at which it can be accessed.

- **AArch32**
  - The execution state is 32 bits.
  - Provides 13 32-bit general-purpose registers, as well as a 32-bit PC, SP, and link register in this execution state (LR). Both an ELR and a procedure link register, the LR is utilized.
  - For use in different PE modes, some of these registers contain numerous banked instances.
  - Provides a single ELR for Hyp mode exception returns.
  - Supports Advanced SIMD vector and scalar floating-point with 32 64-bit registers.
  - Both A32 and T32 instruction sets are included.
  - Supports the ARMv7-A exception model, which is based on PE modes, and translates it to the ARMv8 Exception model, which is based on Exception levels.
  - 32-bit virtual addresses are used.
  - The PE state is stored in a single Current Program State Register (CPSR).
  - Interprocessing is the process of switching between the AArch64 and AArch32 execution states.

Only by changing the Exception level can the PE switch between execution states. This means that software layers executing at distinct Exception levels, such as an application, an operating system kernel, and a hypervisor, might execute in separate execution states.

ii] ARM instruction sets

The potential instruction sets in ARMv8 are determined by the execution state:

- **AArch64:** Only one instruction set, A64, is supported by the AArch64 state. This is a 32-bit instruction set with a fixed length instruction set.
- **AArch32:** The following instruction sets are supported by the AArch32 state:
- **A32:** This is a 32-bit instruction set with a fixed length instruction set. It can be used with the ARMv7 instruction set
- **T32:** This is a variable-length instruction set with both 16-bit and 32-bit encodings. The ARMv7 Thumb® instruction set is supported
- Each of these instruction sets is expanded by ARMv8.

The instruction set that the PE executes is determined by the PE Instruction set state. SIMD and scalar floating-point instructions are supported by the ARMv8 instruction sets.

iii] System registers

Control and status information for architected features are provided through system registers. The naming format for System registers is <register_name>.<bit_field_name> to identify specific registers, as well as control and status bits within a register, use bit field name.

Bits can also be expressed numerically in the form <register name>[x:y] or in the generic form bits[x:y].

In addition, most register names in the AArch64 state include the lowest Exception level that can access the register as a suffix: <register_name>_ELx, where x is 0, 1, 2, or 3

The System registers consists of:

- General system control registers
- Registers for debugging.
- Timer registers that are generic.
- Performance Monitor can optionally register.
- Trace registers are optional.
- Generic Interrupt Controller (GIC) CPU interface registers are optional.

iv] ARMv8 Debug

The following are supported by ARMv8:

- **Debugging on your own server**

  The PE generates debug exceptions in this model. The ARMv8 Exception model includes debug exceptions.

- **Debugging from the outside**

  Debug events cause the PE to enter the Debug state in this model. The PE is managed by an external debugger in the Debug stage.

  Both models are supported by all ARMv8 implementations. The model chosen by a given user is determined by the debug requirements at various phases of the product's design and development life cycle. External debug, for example, may be utilized during hardware implementation and OS bring-up, while self-hosted debug could be used during program development.

1.4.2.5 Supported data types

The following integer data types are supported by the ARMv8 architecture:

- **Byte:** 8 bits
- **Halfword:** 16 bits
- **Word:** 32 bits
- **Doubleword:** 64 bits
- **Quadword:** 128 bits

Floating-point data types such as half precision, single precision, double precision, are also supported by the architecture.

It also supports:

- Fixed-point word and doubleword interpretation.
- Vectors, which consist of numerous elements of the same data type held in a single register.

There are two register files in the ARMv8 architecture:

- A registration file that can be used for general purpose.
- A file with SIMD and floating-point registers.

The available register sizes in each of them are determined by the Execution state.

In AArch64 state:

- A general-purpose register file comprises 64-bit registers in the AArch64 state
  - These registers can be accessed as 64-bit registers or as 32-bit registers by using only the bottom 32 bits in many operations.
- There are 128-bit registers in a SIMD and floating-point register file
  - The quadword integer data types are applicable only to the SIMD and floating-point register files.
  - The floating-point data types are applicable only to the SIMD and floating-point register files. Despite the fact that the AArch64 vector registers provide 128-bit vectors, the effective vector length depends on the A64 instruction encoding utilized.

In AArch32 state:

- A general-purpose register file comprises 32-bit registers in the AArch32 state:
  - A doubleword can be supported by two 32-bit registers.
  - The use of vector formatting is possible.
- 64-bit registers are contained in a SIMD and floating-point register file:
  - The quadword integer and floating-point data types are not supported in the AArch32 state.
  - A 128-bit register is made up of two successive 64-bit registers.

1.4.2.6 ARM memory model

The ARM memory model supports the following:

- Exception generation on an unaligned memory access is supported by the ARM memory model.
- Restricting application access to specific memory locations.
- Converting virtual addresses from executable instructions to physical addresses.
- Switching between big-endian and little-endian interpretation of multi-byte data.
- Managing the order in which memory accesses are made.
- Caches and address translation structures are under control.
- Multiple PEs accessing shared memory at the same time.

Support for virtual addresses (VA) is conditional on the Execution state, as follows:

AArch64 state

The Translation Control Register determines the VA range supported by the AArch64 state, which supports 64-bit virtual addressing. Two distinct VA ranges with their own translation controls are supported by execution at EL1 and EL0.

AArch32 state

The Translation Control Register determines the VA range supported by the AArch32 state, which supports 32-bit virtual addressing. The VA range can be split into two subranges, each with its own translation controls, for execution at EL1 and EL0.

System software can discover the supported physical address space, which is IMPLEMENTATION DEFINED. The Virtual Memory System Architecture (VMSA) can translate VAs to blocks or pages of memory anywhere within the supporting physical address space, regardless of the Execution state.

1.5 INTRODUCTION TO RASPBERRY PI

Raspberry Pi is a series of compact single-board computers developed by the Raspberry Pi Foundation in collaboration with Broadcom in the United Kingdom. These projects are generally inclined towards teaching and promoting basic computer science in schools and in developing countries. Due to its low cost, modularity and open design it finds wide application ranging from weather monitoring, robotics and many more.

Several generations have been released of Raspberry Pi's such as Raspberry Pi Model B (February 2012), followed by Model A, Model B+ (in 2014), Raspberry Pi 2(February 2015), Raspberry Pi Zero (November 2015), Raspberry Pi Zero W (On 28 February 2017), Raspberry Pi Zero WH (On 12 January 2018), Raspberry Pi 3 Model B (February 2016), Raspberry Pi 3 Model B+ (2018), Raspberry Pi 4 Model B (released in June 2019), Raspberry Pi 400 (November 2020) and Raspberry Pi Pico (in January 2021).

1.5.1 Introduction to Raspberry Pi

The Raspberry Pi is a fascinating device: it's a fully functional computer packed into a small and inexpensive compact. Whether you want to use the Raspberry Pi to surf the web or play games, learn how to write your own programs, or build your own circuits and physical devices, the Raspberry Pi – and its incredible community – will be there to help you every step of the way.

The Raspberry Pi is a single-board computer, which means it's a computer that's similar to a desktop, laptop, or smartphone but is built on a single printed circuit board. The Raspberry Pi, like most single-board computers, is little – it has about the same footprint as a credit card – but that doesn't mean it's not powerful: it can accomplish everything a larger, more power-hungry computer can do, just not as rapidly.

The Raspberry Pi family was created out of a desire to promote more hands-on computer education throughout the world. The Raspberry Pi Foundation, which was founded by its designers, had no clue it would become so popular: the first few thousand units manufactured in 2012 to test the waters were quickly sold out, and millions have been distributed all over the world in the years afterwards. These circuit boards have been found in homes, classrooms, businesses, data centers, factories, and even self-driving boats and space balloons.

Since the initial Model B, other Raspberry Pi variants have been released, each with enhanced specifications or functionality tailored to a certain use-case. The Raspberry Pi Zero line, for example, is a miniature version of the full-size Raspberry Pi that foregoes a few capabilities – notably multiple USB ports and a wired network interface – in favor of a much smaller footprint and lower power consumption.

Raspberry Pi is a single-board computer with a compact footprint. The Raspberry Pi may be used as a little computer by adding peripherals such as a keyboard, mouse, and display. Raspberry Pi is a popular platform for real-time image/video processing, IoT applications, and robotics. The Raspberry Pi is slower than a laptop or desktop computer, but it is still a computer that can give all of the expected features and abilities while using very little power.

Raspbian OS is based on Debian and is officially provided by the Raspberry Pi Foundation. They also offer NOOBS OS for Raspberry Pi. Several Third-Party OS versions, such as Ubuntu, Archlinux, RISC OS, Windows 10 IOT Core, and others, can be installed.

Raspbian OS is an approved operating system that may be used for free. This operating system is well-suited to the Raspberry Pi. Raspbian has a graphical user interface (GUI) that provides tools for browsing, Python programming, office, gaming, and more. To save the OS, we should use an SD card (minimum 8 GB is advised) (operating System).

Raspberry Pi is more than a computer because it allows developers to access on-chip hardware, such as GPIOs, to create applications. By using GPIO, we may connect and control devices such as LEDs, motors, and sensors. It includes an ARM-based Broadcom Processor SoC as well as an on-chip GPU (Graphics Processing Unit).

Raspberry Pi's CPU speed ranges from 700 MHz to 1.2 GHz. It also includes SDRAM on board, which varies from 256 MB to 1 GB. On-chip SPI, I2C, I2S, and UART modules are also available for the Raspberry Pi.

The Raspberry Pi is available in a variety of versions, which are 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 following are the features of the aforementioned versions of Raspberry Pi that are most commonly used as described in the Table II:

Table II: Features of various versions of Raspberry Pi

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 Freq.	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	250MHz Videocore IV	400 MHz Videocore IV	250MHz Videocore IV
Storage	micro-SD	Micro-SD	micro-SD	micro-SD
Ethernet	Yes	Yes	Yes	No
Wireless	WiFi and Bluetooth	No	No	No

1.5.1.1 What's the Raspberry Pi foundation?

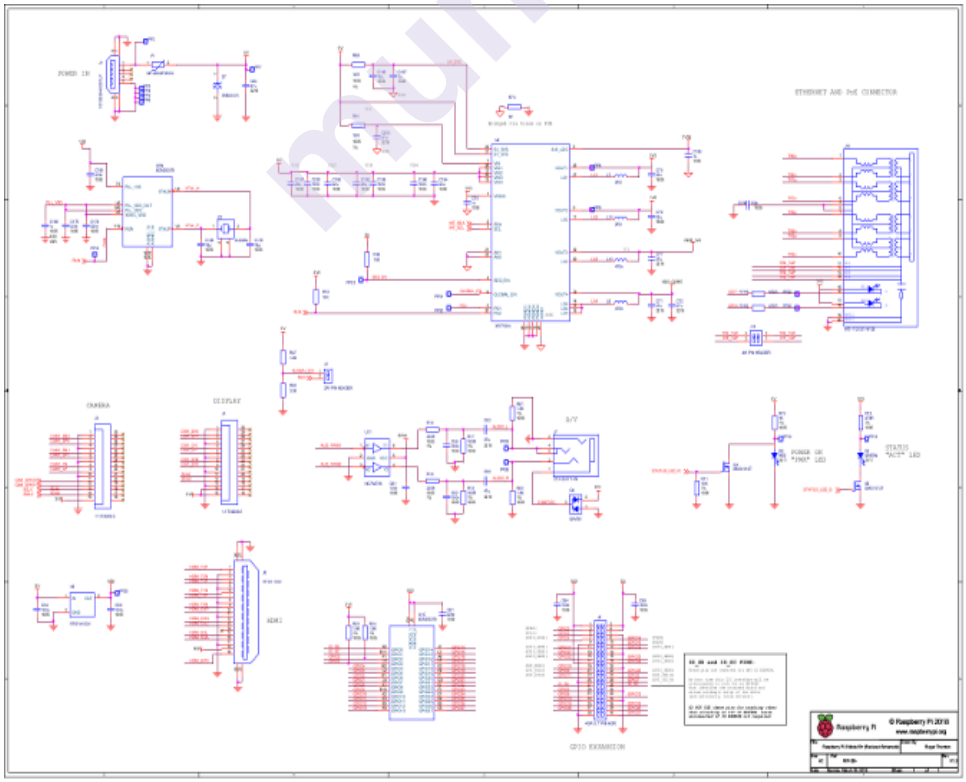
The Raspberry Pi Foundation is dedicated to putting the power of computers and digital fabrication into the hands of people all over the world. It accomplishes this by making low-cost, high-performance computers available for people to learn, solve issues, and have fun with. It conducts outreach and education to assist more people gain access to computing and digital making—it creates free materials to help people learn about computers and how to make things with them, and it also trains educators to help others learn.

The Raspberry Pi Foundation sponsors Code Club and CoderDojo, however both programs are platform-agnostic (they aren't bound to Raspberry Pi hardware). The Raspberry Pi Foundation promotes these clubs and assists in the expansion of the network around the world, ensuring that every child has the opportunity to learn about computers. Raspberry Jams, on the other hand, are Raspberry Pi-focused gatherings where people of all ages can learn about the Raspberry Pi and exchange ideas and projects.

1.5.1.2 Is Raspberry Pi open source?

The Raspberry Pi runs Linux (a number of versions), and its main supported operating system, Pi OS, is open source and runs a suite of open source software. The Raspberry Pi Foundation contributes to the Linux kernel and other open source projects, as well as publishing open source versions of many of its own software.

The schematics for the Raspberry Pi are frequently given as documentation, but the board is not open hardware.



1.5.1.3 Uses of Raspberry Pi

- **Community**

  One of the most intriguing aspects of the project, according to Jamie Ayre of FLOSS software business AdaCore, is the Raspberry Pi community. According to community blogger Russell Davis, the Foundation's strength allows it to focus on documentation and education. The community created The MagPi, a fanzine based on the platform that was handed over to the Raspberry Pi Foundation by its volunteers in 2015 to be maintained in-house. Across the UK and around the world, a series of community Raspberry Jam events have taken place.

- **Education**

  As of January 2012, inquiries about the board had been received from schools in both the public and private sectors in the United Kingdom, with the latter receiving around five times as much interest. Businesses are hoped to finance purchases for less fortunate schools. Premier Farnell's CEO stated that the government of a Middle Eastern country has expressed interest in distributing a board to every schoolgirl in order to improve her employment possibilities.

  The Raspberry Pi Foundation engaged a number of members of its community, including former teachers and software developers, in 2014 to create a set of free instructional tools on its website. The Foundation also launched Picademy, a teacher training program aimed at assisting teachers in preparing to teach the new computing curriculum using the Raspberry Pi in the classroom.

  NASA launched the JPL Open Source Rover Project in 2018 to encourage students and hobbyists to get involved in mechanical, software, electronics, and robotics engineering. The JPL Open Source Rover Project is a scaled-down version of the Curiosity rover that uses a Raspberry Pi as the control module.

- **Home automation**

  The Raspberry Pi is being used by a variety of developers and applications for home automation. These programmers are working to turn the Raspberry Pi into a low-cost energy monitoring and power usage solution. Because of the Raspberry Pi's low price, it has become a popular and cost-effective alternative to more expensive commercial solutions.

- **Industrial automation**

  TECHBASE, a Polish industrial automation company, released ModBerry, an industrial computer based on the Raspberry Pi Compute Module, in June 2014. The device includes a variety of interfaces, including RS-485/232 serial ports, digital and analogue inputs/outputs, CAN, and low-cost 1-Wire buses, all of which are

common in the automation sector. Because of the design, the Compute Module can be utilized in tough industrial conditions, implying that the Raspberry Pi is no longer confined to home and science projects, but can be extensively used as an Industrial IoT solution to fulfill Industry 4.0 goals.

SUSE announced commercial support for SUSE Linux Enterprise on the Raspberry Pi 3 Model B in March 2018, with a handful of unknown customers using the Raspberry Pi to provide industrial monitoring. TECHBASE introduced a Raspberry Pi Compute Module 4 cluster in January 2021 for usage as an AI accelerator, routing, and file server. One or more regular Raspberry Pi Compute Module 4s are housed in an industrial DIN rail enclosure, with some variants including one or more Coral Edge tensor processing units.

- **Commercial products**

  Critter & Guitari designed and manufactured the Organelle, a portable synthesiser, sampler, sequencer, and effects processor. It has a Raspberry Pi computer module that runs Linux on it. Next Thing Co. invented the OTTO digital camera. It has a Raspberry Pi Compute Module built in. It was successfully crowdfunded through a Kickstarter effort in May 2014. Slice is a digital media player that is powered by a Compute Module. It was funded through a Kickstarter effort in August of 2014. Slice's operating system is based on Kodi. The Raspberry Pi is used in a number of commercial thin client computer terminals.

- **Covid-19 pandemic**

  During the coronavirus pandemic in Q1 2020, Raspberry Pi computers saw a significant increase in demand, owing to an increase in working from home, as well as the use of many Raspberry Pi Zeros in ventilators for COVID-19 patients in countries like Colombia, which helped to relieve strain on the healthcare system. Raspberry Pi sales surpassed 640,000 units in March 2020, the second highest month in the company's history.

1.5.1.4 Raspberry Pi foundation hall of fame

Members of the Raspberry Pi Hall of Fame include:

A] Eben Upton

Eben Christopher Upton is presently employed by Broadcom as a Technical Director and ASIC Architect. He is the man who is known for being the founder and former trustee of the Raspberry Pi Foundation, as well as the current CEO of the Raspberry Pi trading firm. Eben Upton's primary responsibility is the creation of the Raspberry Pi device's general software and hardware architecture.

B] Paul Beech

Paul Beech designed the current Raspberry Pi Foundation logo and is now working on producing diagrams, posters, and developing the Official Raspberry Pi website. Pimoroni, which makes Pibow, PiHub, Pibrella, and other useful doo-hickeys to make raspberry pi more fun to study and engage with, counts him as a founding member.

C] Alex Bradbury

Alex Bradbury, a Ph.D. student at the University of Cambridge, has been a volunteer for the Raspberry Pi Foundation since its inception. Alex is in charge of maintaining repositories that contain custom versions of the Raspbian operating system, and he has even co-authored a popular book titled "Learning Python with Raspberry Pi."

D] Dom Cobley

Dom Cobley (Engineer at Broadcom) has made a number of successful contributions to turning the Raspberry Pi into a Media Streaming device. Dom Cobley has provided VideoCore firmware for the Raspberry Pi, Kernel maintenance, and even developed XBMC (Xtreme Box Media Center) as a developer to enable media streaming (Video and Audio) over the Raspberry Pi.

E] Peter Green

Peter Green created the Raspbian Debian derivative and manages the Raspbian repository. Peter is now working on making a stable Raspbian version based on Debian Jessie (Debian 8) available.

F] James Hughes

Since 2011, James Hughes has been one of the original volunteers for the Raspberry Pi Foundation. He is now the chief developer of Camera Board Software, the Moderator of the Pi Forum, and the diligent maintainer of the Raspberry Pi website and Twitter page.

G] Mike Thompson

Mike Thompson collaborated on the Raspbian operating system for the Raspberry Pi alongside Peter Green.

H] Gert Van Loo

Gert Van Loo is a Broadcom engineer who was responsible for the development of the first hardware design of Alpha boards in 2011, which later became known as the "Raspberry Pi." In addition, he created the Gertboard and Gertduino expansion boards for the Raspberry Pi.

I] Rob Mullins

Along with Eben Upton, Rob Mullins was a co-founder of the Raspberry Pi Foundation and served as a trustee until 2014. He is currently employed as a Senior Lecturer in the University of Cambridge's Computer Laboratory. Computer Architecture and VLSI- On-chip Interconnection networks, chip-multi-processors, and innovative parallel processing fabrics are among his areas of specialization.

1.5.1.5 Advantages of Raspberry Pi

The following are the benefits of using a Raspberry Pi:

1. The Raspberry Pi is a small, powerful, and efficient cum compact form factor computer that is also quite inexpensive to purchase. Raspberry Pi can be used by a variety of small and medium-sized businesses to perform functions such as web server, database server, and media server. As a result, a significant amount of money can be saved on the purchase of numerous servers.
2. Raspberry Pi can be used as a single platform for a wide range of programming tasks. Pi supports a variety of programming languages, and users can install the appropriate compiler to ensure proper code execution. Python, the main programming language used by Pi, is a simple and easy-to-learn language. It allows for more efficient code creation, fewer lines of code, and automatic memory management.
3. The product is open source and supports open source operating systems and apps. As a result, Raspberry Pi has access to a large number of operating systems in various variants of Linux, as well as millions of apps for that operating system.
4. The Raspberry Pi includes add-on hardware such as the Camera, Component Moduler Kit, Gertboard, and HAT board, allowing users to connect thousands of third-party devices like as buttons and LEDs to perform various tasks on the Pi.
5. The product is energy efficient and offers small businesses a greener, more ethical option. This credit card-sized product is simple to recycle and saves money on cooling solutions.

1.5.1.6 Disadvantages/Limitations of Raspberry Pi

The following are the Raspberry Pi's limitations/drawbacks:

1. Because the Ethernet Port and Processing CPU are not fast enough to process multitasking computing cycles, it cannot function as a full-fledged computer.
2. Does not work with a fully functional Windows operating system.
3. The product is limited to SMEs and is not particularly beneficial, whereas larger organizations/enterprises have access to a wide range of facilities.

4. Doesn't have a battery-backed Real Time Clock (RTC). NTP Server is the sole way to work with time, and most operating systems do this automatically.

5. There is no built-in ADC converter. For ADC, an external charger is used.

6. Bluetooth and Wi-Fi are not supported out of the box, and numerous USB-based dongles are likewise not supported for wireless connectivity.

1.5.1.7 Generations of Raspberry Pi

Various generations of Raspberry Pi, ranging from the Raspberry Pi Model A through the Raspberry Pi Model B+, as well as the recently introduced Raspberry Pi Zero, will be addressed in this part, along with their extensive technical specifications.

Raspberry Pi Model A

The Raspberry Pi Model A is the first generation of Raspberry Pi models to be launched. Two models of the Raspberry Pi Model A were released: the Raspberry Pi 1 Model A and the Raspberry Pi 1 Model A+.

A] Raspberry Pi: Model A

The Raspberry Pi Model A as shown in the figure 1.10 is a lower-spec version of the Raspberry Pi. Because this model of Pi lacks crucial hardware interfaces, it was designed specifically for embedded projects. In comparison to Model B, Model A is lighter and uses less energy. Model A has become obsolete and is no longer easily accessible on the market.

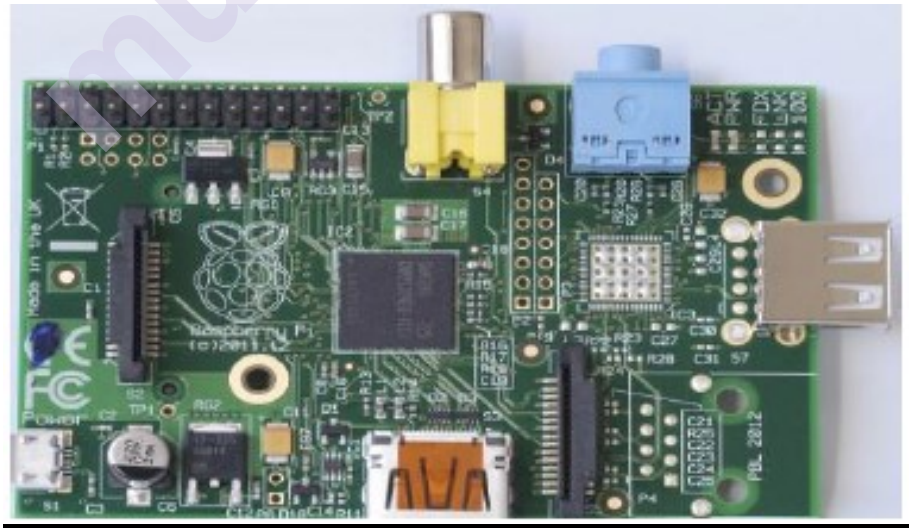


Figure 1.10 Raspberry Pi: Model A

The Technical Specification of the Raspberry Pi 1 Model A is listed in the Table III below:

Table III: Specifications of Raspberry Pi: Model A

Hardware parameters	Description
SoC	Broadcom BCM2835
CPU	700 MHz Single Core ARM 1176JZF-S
GPU	Broadcom VideoCore IV @ 250 MHz
RAM	256 MB
Onboard Ports	1 USB; 1 HDMI (Ver 1.4); 3.5mm Sound Jack
Video Input	15-pin MIPI camera interface (CSI) Connector
Audio Input	2 Boards via I2S
Onboard Storage	SD/MMC/SDIO Card slot
Ethernet	No
GPIO	8 GPIO including UART, I2C, SPI Bus with two chip selects, I2S audio, +3.3 V, +5V, GND
Adapter Rating	5V; 300 mA
Launch Date	February 2013
Price	$25

B] Raspberry Pi: Model A+

In terms of size and power consumption, the Raspberry Pi 1 Model A+ as illustrated in the figure 1.11 was the successor and well-updated model of the Raspberry Pi 1 Model A. Additional GPIO pins, MicroSD card capability, and better audio reproduction are among the enhancements made with the Model A+. The Raspberry Pi Model A+ could also run a variety of operating systems and serve as a solid backbone for a variety of space-related applications and media center operations. Because more advanced variants are now available, the Model A+ is likewise being phased out of the market.

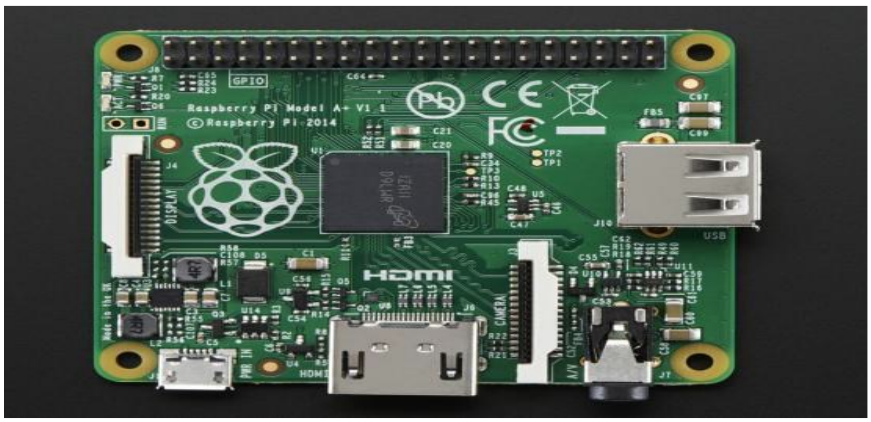


Figure 1.11 Raspberry Pi: Model A+

The Technical Specification of the Raspberry Pi 1 Model A+ is listed in the table IV below:

Table IV: Specifications of Raspberry Pi: Model A+

Hardware parameters	Description
SoC	Broadcom BCM2835
CPU	700 MHz Single Core ARM 1176JZF-S
GPU	Broadcom VideoCore IV @ 250 MHz
RAM	256 MB
Onboard Ports	1 USB; 1 HDMI (Ver 1.4); 3.5mm Sound Jack
Video Input	15-pin MIPI camera interface (CSI) Connector
Audio Input	2 Boards via I2S
Onboard Storage	MicroSD Card slot
Ethernet	No
GPIO	17 GPIO including UART, I2C, SPI Bus with two chip selects, I2S audio, +3.3 V, +5V, GND, HAT ID Bus
Adapter Rating	5V; 200 mA
Launch Date	February 2014
Price	$20

C] Raspberry Pi: Model B

Because of the large RAM, additional USB port slots, and Ethernet port, the Raspberry Pi 1 Model B (Figure 1.12) was viewed as a higher specification model of the Pi 1 Model A with good working performance. Raspberry Pi 1 Model B paved the way for children to pursue computing as a hobby, leading to education, programming, and home projects.

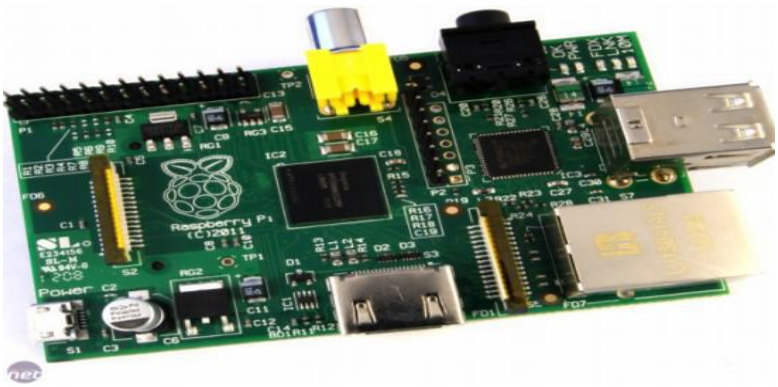


Figure 1.12 Raspberry Pi: Model B

The Technical Specification of the Raspberry Pi 1 Model B is listed in the table V below:

Table V: Specifications of Raspberry Pi: Model B

Hardware parameters	Description
SoC	Broadcom BCM2835
CPU	700 MHz Single Core ARM 1176JZF-S
GPU	Broadcom VideoCore IV @ 250 MHz
RAM	512 MB
Onboard Ports	2 USB; 1 HDMI (Ver 1.4); 3.5mm Sound Jack
Video Input	15-pin MIPI camera interface (CSI) Connector
Audio Input	2 Boards via I2S
Onboard Storage	SD/MMC/SDIO Card
Ethernet	10/100 Mbps
GPIO	8 GPIO including UART, I2C, SPI Bus with two chip selects, I2S audio, +3.3 V, +5V, GND, Additional 4 GPIO on P5 pad
Adapter Rating	5V; 700 mA
Launch Date	February 2012
Price	$35

C] Raspberry Pi: Model B+

Under the Raspberry Pi 1 models category, Model B+ (Figure 1.13) was considered the last cum final version. Model B+ superseded Model B and had more enhanced hardware features like as more GPIO, more USB ports, a better MicroSD card, lower power consumption, and better audio output when compared to all Raspberry 1 generations products.

Figure 1.13 Raspberry Pi: Model B+

The Technical Specification of the Raspberry Pi 1 Model B+ is listed in the table VI below:

Table VI: Specifications of Raspberry Pi: Model B+

Hardware parameters	Description
SoC	Broadcom BCM2835
CPU	700 MHz Single Core ARM 1176JZF-S
GPU	Broadcom VideoCore IV @ 250 MHz
RAM	512 MB
Onboard Ports	4 USB; 1 HDMI (Ver 1.4); 3.5mm Sound Jack
Video Input	15-pin MIPI camera interface (CSI) Connector
Audio Input	2 Boards via I2S
Onboard Storage	MicroSD Card
Ethernet	10/100 Mbps
GPIO	17 GPIO including UART, I2C, SPI Bus with two chip selects, I2S audio, +3.3 V, +5V, GND, HAT ID bus
Adapter Rating	5V; 600 mA
Launch Date	July 2014
Price	$25

Raspberry Pi 2 Model B

After the Model A generations, the Raspberry Pi Model B generations were released, with improved functionality, more powerful hardware, and better operating system support.

A] Raspberry Pi 2: Model B

The Raspberry Pi 2 Model B (Figure 1.14) is the Raspberry Pi's second iteration. In terms of a powerful CPU, RAM, GPIO, and other hardware connector features, it superseded the Raspberry Pi 1 Model B+ variants.

Figure 1.14 Raspberry Pi2: Model B

The Technical Specification of the Raspberry Pi 2 Model B is listed in the table VII below:

Table VII: Specifications of Raspberry Pi2: Model B

Hardware parameters	Description
SoC	Broadcom BCM2836
CPU	900 MHz Quad-Core ARM Cortex-A7
GPU	Broadcom VideoCore IV @ 250 MHz
RAM	1 GB
Onboard Ports	4 USB; 1 HDMI (Ver 1.4); 3.5mm Sound Jack
Video Input	15-pin MIPI camera interface (CSI) Connector
Audio Input	2 Boards via I2S
Onboard Storage	MicroSD Card
Ethernet	10/100 Mbps
GPIO	17 GPIO including UART, I2C, SPI Bus with two chip selects, I2S audio, +3.3 V, +5V, GND, HAT ID bus
Adapter Rating	5V; 800 mA
Launch Date	February 2015
Price	$35

Raspberry Pi - Zero

The Raspberry Pi ZERO (Figure 1.15) is a new member of the Raspberry Pi family. It is the cheapest and most affordable board, costing around $5. Raspberry Pi Zero is capable of running Raspbian and all other programs that other Pi's can. The size is approximately half that of the A+ model, and the quantity of utilities is doubled.

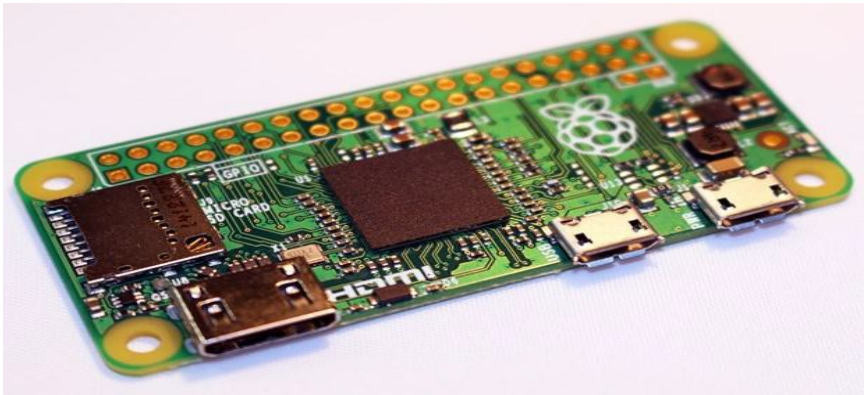


Figure 1.15 Raspberry Pi - Zero

The Technical Specification of the Raspberry Pi Zero is listed in the table VIII below:

Table VIII: Specifications of Raspberry Pi Zero

Hardware parameters	Description
SoC	Broadcom BCM2835
CPU	1GHz ARM 1176JZF-S Single Core
GPU	Broadcom VideoCore IV @ 250 MHz
RAM	512 MB
Onboard Ports	Micro-USB; Mini-HDMI (Ver 1.4); Audio via PWN on GPIO
Video Input	N/A
Audio Input	2 Boards via I2S
Onboard Storage	MicroSD Card
Ethernet	N/A
GPIO	40 GPIO Pins
Adapter Rating	5V; 160 mA
Launch Date	November 2015
Price	$5

1.5.1.8 Raspberry Pi operating systems

The operating system is considered to be the most important software for computer hardware to function and to provide an interface between the computer hardware and the programs that are running. There are numerous operating systems for the Raspberry Pi that are based on Linux and are free and open source.

Raspberry Pi and Linux

Linus Torvalds, the creator of the Linux operating system, made Linux available as a platform for community development. The Raspberry Pi Foundation opted to include Raspbian Pi, an official Linux distribution that is tailored for Raspberry Pi.

Firmware and Kernel

Kernel is regarded as the "Brain" of the whole system, with the operating system serving as the "Outer Body." Kernel is an operating system

component that interacts with installed hardware devices. Because it is software that is semi-permanently written on Partition 1 of the SD card, kernel is also known as "Firmware." This section will provide an overview of the numerous operating systems that can be installed and supported by the Raspberry Pi.

There are two types of operating systems available for the Raspberry Pi:

A] Officially available operating systems:

1] Raspbian Operating System

Based on Debian, the Raspbian operating system is tailored for Raspberry Pi devices. Raspbian is a collection of programmes and utilities that run on the Raspberry Pi. It comes with over 35000 packages and is straightforward to install on the Raspberry Pi.

Raspbian Pi, as the Pi's primary operating system, has been designed for speed and stability, and is also being actively developed by the open source community.

Download: https://www.raspbian.org/RaspbianImages

Latest Version: Raspbian Jessie; Kernel Version: 4.1

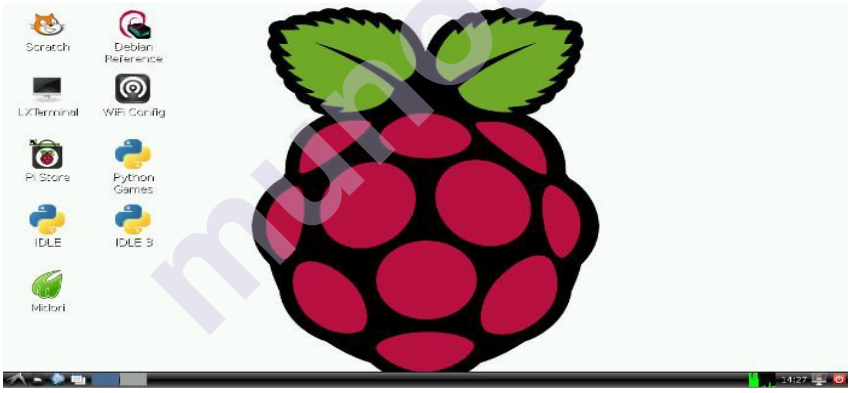


Figure 1.16 GUI interface of Raspbian operating system

2] Arch Linux ARM

Arch Linux ARM is a Linux distribution that is specifically designed for ARM processors. Arch Linux ARM is known for being easy to use and giving end users complete control. It provides a lightweight foundation framework that allows users to configure the system according to their needs, and it is only because of this that Arch Linux ARM lacks a GUI interface.

Arch, like Raspbian, is under constant development and is updated on a regular basis.

Download: https://www.archlinux.org/download/

Latest Version: 2015.12.1

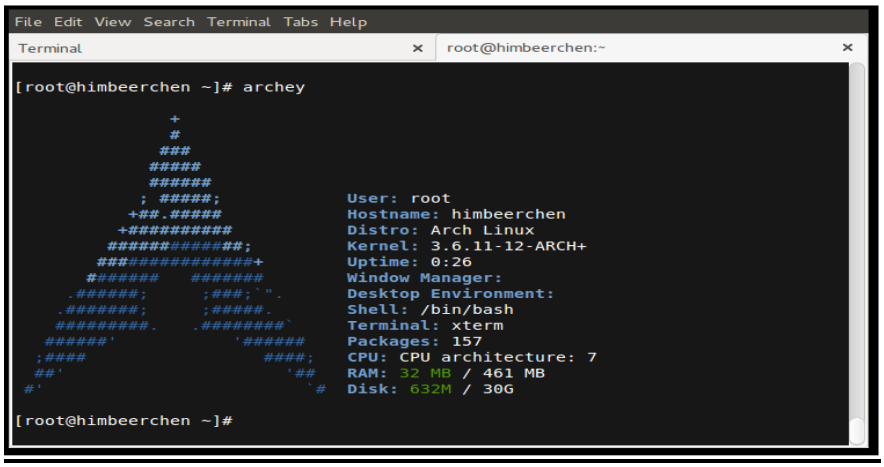


Figure 1.17 CUI Interface of ARCH Linux for Raspberry Pi

3] OpenELEC (Open Embedded Linux Entertainment Center)

OpenELEC is a Linux-based distribution that is specifically built for managing HTPCs and is based on KODI (XBMC-Media Player).

XBMC Frodo 12.1 is supported by OpenELEC. OpenELEC is primarily intended for speedier system booting, and it can transform any blank PC into a full-fledged media streaming computer in about 15 minutes. The OpenELEC operating system is optimised for a variety of architectures, including Atom, ION, Intel, Fusion, Raspberry Pi, and others.

Download: http://openelec.tv/

Latest Version: 3.0.0

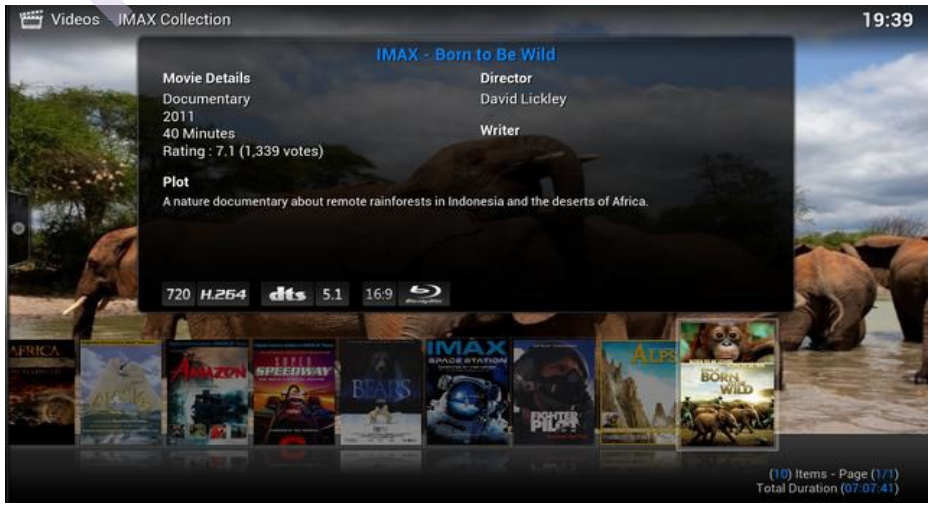


Figure 1.18 GUI Interface of OpenELEC Raspberry Pi

4] Pidora

The “Raspberry Pi Fedora Remix” operating system is also known as Pidora. Pidora is a Linux distro created specifically for the Raspberry Pi. It consists of Fedora Project software packages that have been specifically tailored/modified to work on the Raspberry Pi. Pidora, like a Fedora-based operating system, also provides a platform for the open source community to submit apps to the operating system.

Download: http://pidora.ca/

Latest Version: Pidora 2014

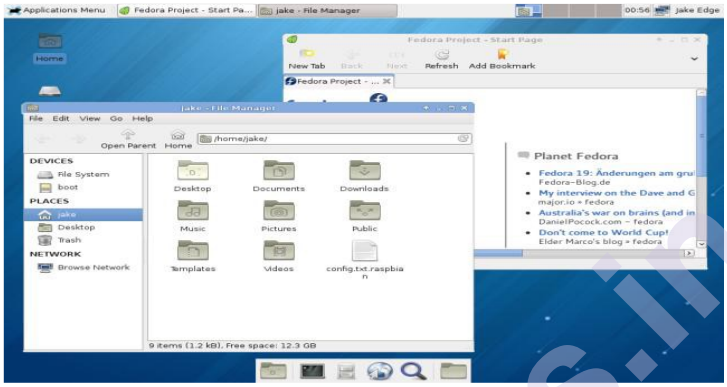


Figure 1.19 GUI Interface of Pidora Operating System

5] Puppy Linux

Puppy Linux is a lightweight distribution that focuses on ease of use and minimal memory use. Puppy Linux has been modified for the Raspberry Pi and includes a large number of application suites. Like other distributions, open source community developers and even penetration testers are working throughout the world to improve the system's reliability, performance, and efficiency, and the community provides regular software and updates for the operating system, as well as bug fixes.

Download:
http://puppylinux.org/main/Download%20Latest%20Release.htm

Latest Version: Slacko Puppy 6.3

Figure 1.20 GUI Interface for Puppy Linux for Raspberry Pi

6] RISC OS

The ARM Team created RISC OS specifically for ARM processors. Because it is not tied to Windows or Linux, RISC OS is an extremely fast, compact, and efficient operating system. It includes a full desktop environment as well as a library of applications for Raspberry Pi.

Download:
https://www.riscosopen.org/content/downloads/raspberry-pi

Latest Version: RISC OS 14

Figure 1.21 GUI Interface of RISC OS for Raspberry Pi

7] OSMC (Open Source Media Center)

Open Source Media Center is a Linux distribution centred on a free and open source media player with over 30000 packages. OSMC is a free and open source operating system that just takes a few minutes to set up. OSMC has a thriving community that releases updates and new packages on a monthly basis. “As OSMC says, ‘Play Anything from Anywhere.”

Download: https://osmc.tv/download/

Version: 2015.11.1

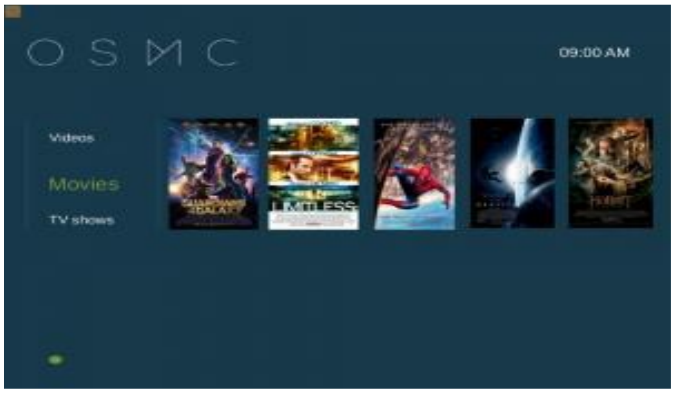


Figure 1.22 GUI Interface of OSMC for Raspberry Pi

8] Ubuntu Mate

Ubuntu Mate is a Raspberry Pi-specific version of Ubuntu 15.10 that was launched on October 22, 2015. With Ubuntu Mate, the Pi now has access to the same huge software repository as Ubuntu. Ubuntu Mate is a full-featured desktop environment that can run a variety of graphical applications as well as other standard Ubuntu tasks. Ubuntu Mate is the result of Raspberry Jams' efforts to improve “out of the box” GPIO functionality.

Download: https://ubuntu-mate.org/wily/

Latest Version: Ubuntu Mate 15.10

Figure 1.23 GUI Interface for Ubuntu Mate 15.10

9] Window 10 IoT core

Microsoft's Windows 10 IoT Core is a platform for creating IoT-based applications for the Raspberry Pi. The Windows 10 IoT core delivers the power of Windows to Raspberry Pi, making it simple to integrate rich experiences such as natural user interfaces, searching, online storage, and even cloud computing with gadgets.

Download: http://ms-iot.github.io/content/en-US/Downloads.htm

Latest Version: Windows 10

Figure 1.24 Windows 10 IoT Core

B] Miscellaneous Operating system

Other operating systems that can be downloaded and installed on the Raspberry Pi include as follows:

1] Q4OS: Raspberry Pi operating systems that are fast and powerful, with a focus on security, dependability, long-term stability, and cautious incorporation of validated new features.

2] Xbian: Xbian is a media centre distribution for the Raspberry Pi that is tiny, fast, and lightweight. Based on Debian minor, this is the fastest Kodi solution for a variety of small form factor computers.

3] openSUSE: SuSE Linux Professional, previously known as SUSE Linux, is a good platform for open source tools for software developers and administrators, as well as a user-friendly desktop and feature-rich server GUI interface.

4] FressBSD: Since November 2012, FreeBSD has supported the Raspberry Pi and is identical to Linux. FreeBSD is a full-featured operating system that includes kernel, device drivers, userland utilities, and documentation.

5] Kali Linux: Kali Linux, a Debian-based forensics and penetration testing operating system, now includes support for the Raspberry Pi. It comes with over 600 testing programs, a graphical user interface, and other ethical hacking tools.

6] SailPi: The SailPi operating system is based on the Sailfish OS 2.0.0.10 version. This operating system has a more powerful OS core, supports a variety of architectures, including Intel Atom and the Raspberry Pi, and offers good security, multitasking, and a better user interface.

1.5.2 Raspberry Pi Hardware

Unlike a standard computer, which has all of its components, ports, and features hidden behind a cover, a Raspberry Pi has all of its components, ports, and functions on show — though you may purchase a case for added protection if you like. This makes it an excellent tool for learning about the functions of various computer components, as well as for figuring out where to plug in the numerous extras (known as peripherals) you'll need to get started. Figure 1.25 (below) depicts the Raspberry Pi from above.

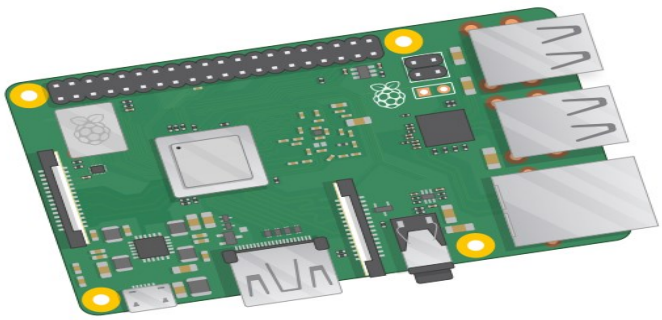


Figure 1.25 Raspberry Pi Model B+

While the Raspberry Pi appears to have a lot crammed (crowd) into its little board, it's actually quite simple to understand, starting with its components and the inner workings that keep the gadget running.

The Pi, like any computer, is made up of a variety of components, each of which plays an important function in its operation. The first, and possibly most essential, of these is the system-on-chip, which can be found right above the center point on the top side of the board (Figure 1.26), covered in a metal cap (SoC).

Figure 1.26 The Raspberry Pi’s system-on-chip

The name system-on-chip gives you a good idea of what you'll find if you pry the metal cover off the Raspberry Pi: a silicon chip, also known as an integrated circuit that houses the majority of the Raspberry Pi's system. This includes the central processing unit (CPU), which is known as a computer's "brain," and the graphics processing unit (GPU), which is in charge of the visual side of things.

However, a brain is useless without memory, and on the Raspberry Pi's underbelly, you'll find just that: another chip, which looks like a small black

plastic square (Figure 1.27). These components work together to create the Pi's volatile and non-volatile memories: the volatile RAM loses its contents when the Pi is turned off, whilst the non-volatile microSD card preserves its contents.

Figure 1.27 Raspberry Pi's random access memory (RAM)

When you flip the board over, you'll notice another metal lid in the upper-right corner, this one with an etched Raspberry Pi logo (Figure 1.28). This section discusses the radio, which allows the Raspberry Pi to communicate wirelessly with other devices. In actuality, the radio has two main functions: a WiFi radio for connecting to computer networks, and a Bluetooth radio for connecting to peripherals such as mice and sending and receiving data from nearby smart devices such as sensors and smartphones.

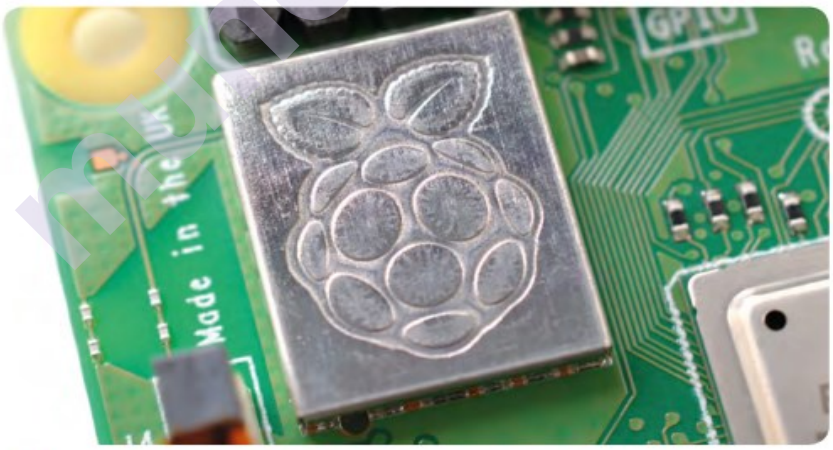


Figure 1.28 The Raspberry Pi's radio module

Just behind the middle row of USB ports, another black, plastic-covered chip can be seen near the bottom border of the board. This is the network and USB controller, which is in charge of the Ethernet port as well as the four USB ports. A final black chip, much smaller than the others, can be found just above the micro USB power connector on the upper-left side of the board (Figure 1.29); this is known as a power management integrated circuit (PMIC), and it handles converting the power from the micro USB port into the power the Pi requires to run.

Figure 1.29 Raspberry Pi's power management integrated circuit (PMIC)

The Raspberry Pi's port

The Raspberry Pi features a variety of ports, starting with four USB ports on the center and right-hand sides of the bottom edge (Figure 1.30). These ports allow you to attach any USB-compatible peripheral to the Pi, including keyboards, mouse, digital cameras, and flash drives. These are known as USB 2.0 ports in technical terms, which indicates they are based on the Universal Serial Bus standard version two.

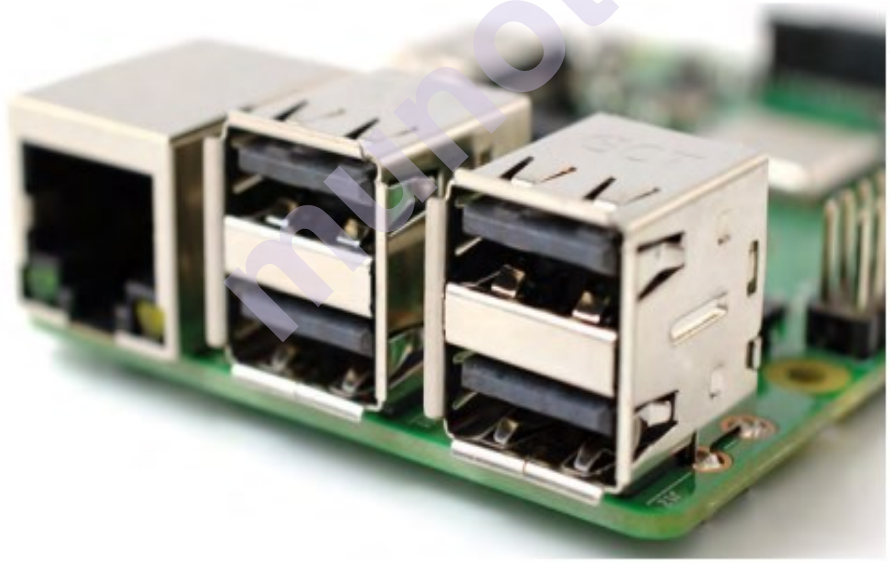


Figure 1.30 The Raspberry Pi's USB ports

An Ethernet port, often known as a network port, is located to the left of the USB ports (Figure 1.31). This port can be used to connect the Raspberry Pi to a wired computer network through a cable with an RJ45 connector on the other end. If you look closely at the Ethernet port, you'll notice two light-emitting diodes (LEDs) on the bottom; these are status LEDs that indicate whether or not the connection is active.

Figure 1.31 The Raspberry Pi's Ethernet ports

A 3.5 mm audio-visual (AV) jack is located just above the Ethernet port on the Raspberry Pi's left-hand edge (Figure 1.32). This is also known as the headphone jack, and it can be used for that purpose — albeit connecting it to amplified speakers rather than headphones will provide superior sound. The 3.5 mm AV jack, however, has a secret feature: in addition to audio, it transmits a video signal that can be linked to TVs, projectors, and other displays that support a composite video signal with a special connection known as a tip-ring-ring-sleeve (TRRS) adapter.

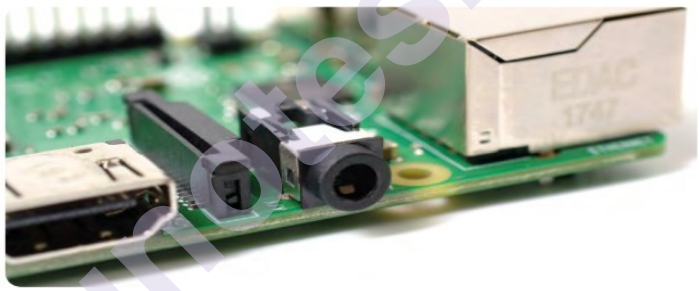


Figure 1.32 The Raspberry Pi's 3.5mm AV jack

A strange-looking connector with a plastic flap that can be pushed up sits directly above the 3.5 mm AV jack; this is the camera connector, also known as the Camera Serial Interface (CSI) (Figure 1.33). This enables you to use the Raspberry Pi Camera Module, which was created specifically for the Raspberry Pi.

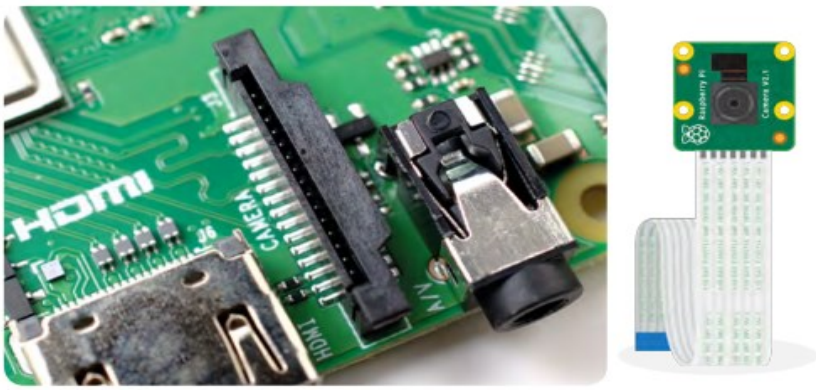


Figure 1.33 Raspberry Pi's camera connector

The High-Definition Multimedia Interface (HDMI) connection (Figure 1.34), which is the same sort of connector seen on a games console, set-top box, and TV, is located above that, still on the left-hand edge of the board. The multimedia component of its name indicates that it can carry both audio and video information, while high-definition indicates that the quality will be superb. This will be used to link the Raspberry Pi to your display device, which could be a computer monitor, television, or projector.

Figure 1.34 Raspberry Pi's HDMI port

A micro USB power port (Figure 1.35), located above the HDMI port, is used to connect the Raspberry Pi to a power source. On smartphones, tablets, and other portable gadgets, the micro USB port is a regular appearance. So you could use a regular phone charger to power the Pi, but the official Raspberry Pi USB Power Supply is recommended for optimum performance.

Figure 1.35 The Raspberry Pi's micro USB power port

Another strange-looking connector (Figure 1.35) can be seen towards the top edge of the board, which at first glance appears to be identical to the camera connector. This, on the other hand, is a display connector, or Display Serial Interface (DSI), made specifically for the Raspberry Pi Touch Display (Figure 1.36).

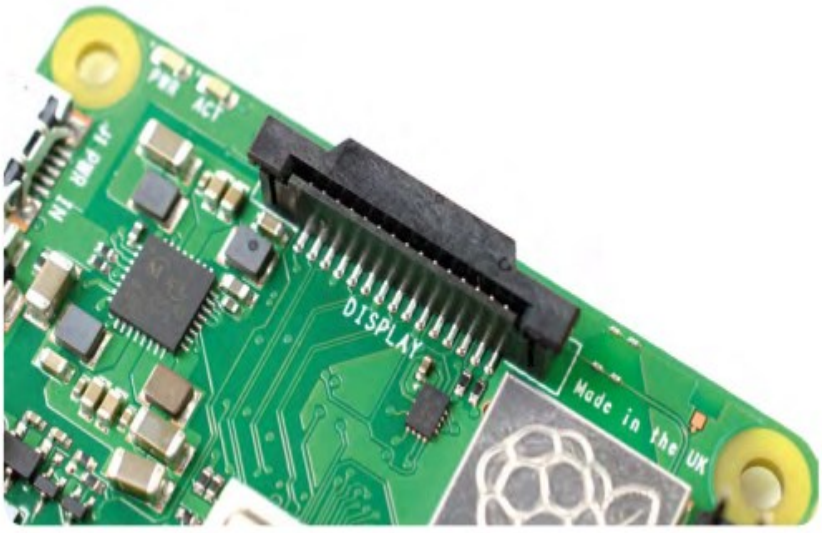


Figure 1.35 The Raspberry Pi's display connector (DSI)

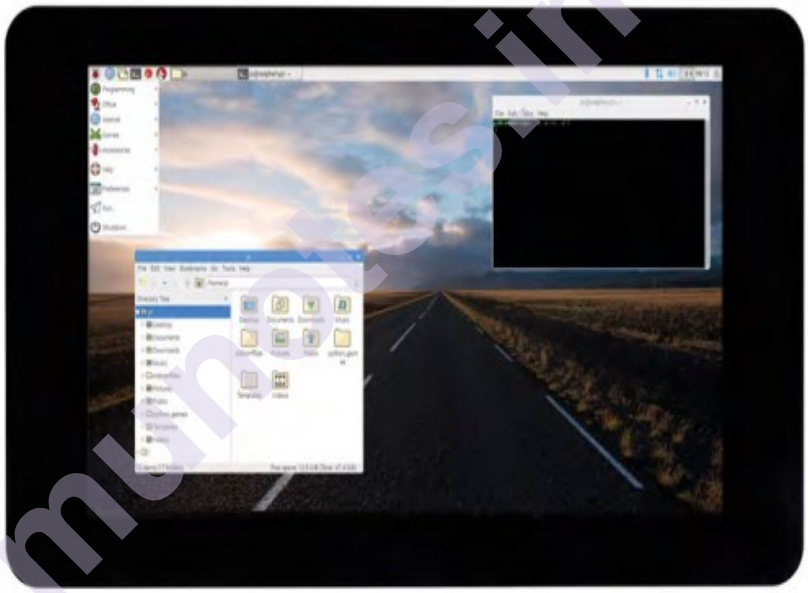


Figure 1.36 The Raspberry Pi's touch display

There are 40 metal pins on the right-hand edge of the board, divided into two rows of 20 pins each (Figure 1.37). The GPIO (general-purpose input/output) header is a feature of the Raspberry Pi that allows it to communicate with external hardware such as LEDs, buttons, temperature sensors, joysticks, and pulse-rate monitors. Another, smaller header with four pins is just below and to the left of this header: This is used to attach the Power over Ethernet (PoE) HAT, an optional add-on that allows the Raspberry Pi to get power through a network connection instead of the micro USB socket.

Figure 1.37 The Raspberry Pi's GPIO header

The Raspberry Pi has one more port, but it's not visible from the top. Turn the board over, and on the opposite side of the board from the display connector is a microSD card connector (Figure 1.38). The Raspberry Pi's storage is as follows: All of the files you save, all of the software you install, and the operating system that makes the Raspberry Pi function are stored on the microSD card put in this slot.

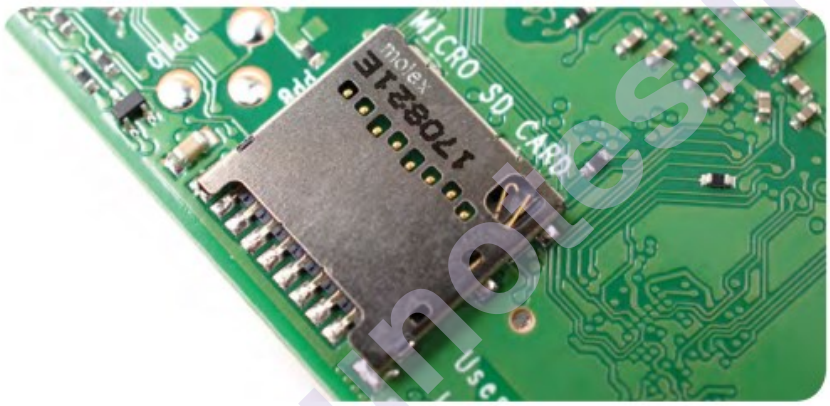


Figure 1.38 The Raspberry Pi's microSD card connector

The Raspberry Pi's peripherals

A Raspberry Pi by itself can't do much, like a desktop computer by itself isn't much more than a door-stop. A microSD card for storage, a monitor or TV to view what you're doing, a keyboard and mouse to tell the Pi what to do, and a 5 volt (5 V) micro USB power source rated at 2.5 amps (2.5 A) or better are all required for the Raspberry Pi to work. You've got yourself a completely functional computer with those.

The Raspberry Pi Case helps protect the Pi while you're using it without blocking access to its various ports; the Camera Module, the Raspberry Pi Camera Module; the Raspberry Pi Touch Display, which connects to the display port and provides both a video display and a tablet-style touchscreen interface; and the SPI Display, which connects to the display port and provides both a video display and a tablet-style touchscreen interface; the SPI Display and the sense HAT (figure 1.39)

A wide range of third-party accessories are also available, ranging from kits to convert a Raspberry Pi into a laptop or tablet to add-ons that allow it to comprehend and respond to your voice.

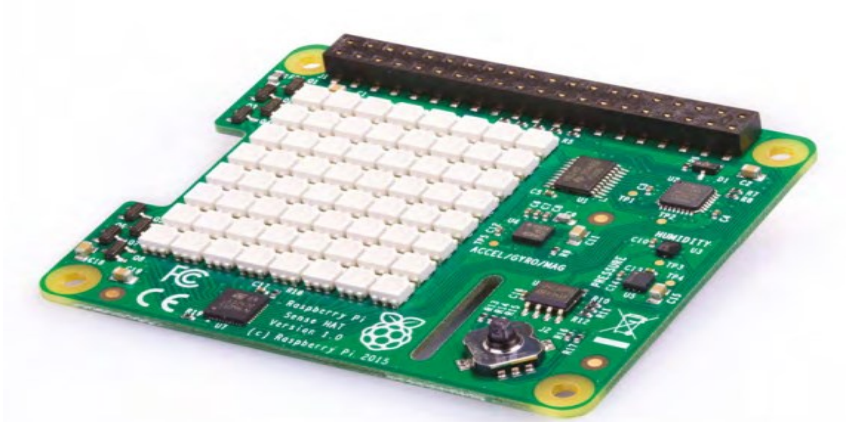


Figure 1.39 The sense HAT

1.5.3 Preparing your raspberry Pi

The Raspberry Pi was created to be as simple to set up and operate as possible, but it, like any computer, is dependent on a variety of external components known as peripherals. While it's fine to look at the Raspberry Pi's bare circuit board, which differs drastically from the encased, closed-off computers you're used to, and worry that things are about to get complicated, this isn't the case. Simply follow the procedures outlined in the next section to have the Raspberry Pi up and running in under ten minutes.

If you have the Raspberry Pi Starter Kit, you'll have almost everything you need to get started: all you need is a computer monitor or TV with an HDMI connection (the same type of connector used by set-top boxes, Blu-ray players, and games consoles) so you can see what the Raspberry Pi is up to. If you don't have the Raspberry Pi Starter Kit, you'll also need the following items in addition to the Raspberry Pi 3 Model B+:

USB power supply: A power supply having a micro USB connector and a rating of 2.5 amps (2.5A) or 12.5 watts (12.5W). The Official Raspberry Pi Power Supply is the best option because it can handle the Raspberry Pi's fast switching power demands.

NOOBS on a microSD card: The microSD card serves as the Raspberry Pi's permanent storage, storing all of the data you generate and software you install, as well as the operating system itself. A 8GB card will get you started, but a 16GB card will give you more room to expand. Using a card with pre-installed NOOBS (New Out-Of-Box Software) will save your time.

USB keyboard and mouse: The Raspberry Pi can be controlled with the help of a USB keyboard and mouse. Almost any USB-connected wired or wireless keyboard and mouse will function with the Raspberry Pi, though

some ‘gaming' keyboards with colourful LEDs may drain too much power to be used reliably.

HDMI Cable: The HDMI cable connects your Raspberry Pi to your TV or monitor and transmits sound and video. There's no need to splurge on a high-end HDMI cable. If you want to connect your Raspberry Pi to an older TV that uses composite video or has a SCART socket, use a 3.5 mm tip-ring-ringsleeve (TRRS) audio/video cable; if you want to connect your Raspberry Pi to an older TV that uses composite video or has a SCART socket, use a 3.5 mm tip-ring-ringsleeve (TRRS) audio/video cable.

Without a case, the Raspberry Pi is safe to use as long as it is not placed on a metal surface that could conduct electricity and cause a short-circuit. However, an optional case can give further protection; the Starter Kit contains the Official Raspberry Pi Case, while third-party cases can be found at any respectable retailer.

You'll also need a network cable if you wish to use the Raspberry Pi on a wired network rather than a wireless (WiFi) network. This should be connected to your network's switch or router on one end. You won't need a cable if you wish to utilize the Raspberry Pi's built-in wireless radio; you will, however, need to know the name and key or pass for your wireless network.

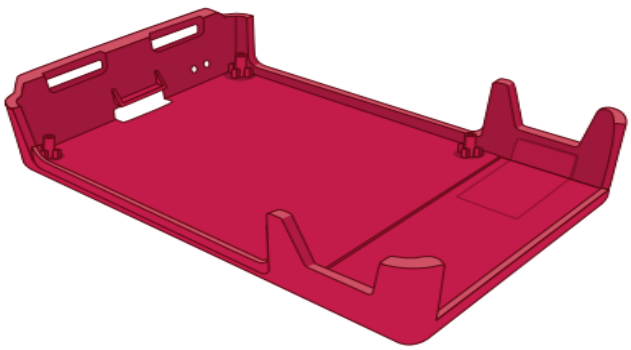
1.5.3.1 Setting up the hardware

You'll also need a network cable if you wish to use the Raspberry Pi on a wired network rather than a wireless (WiFi) network. This should be connected to your network's switch or router on one end. You won't need a cable if you wish to utilize the Raspberry Pi's built-in wireless radio; you will, however, need to know the name and key or pass for your wireless network.

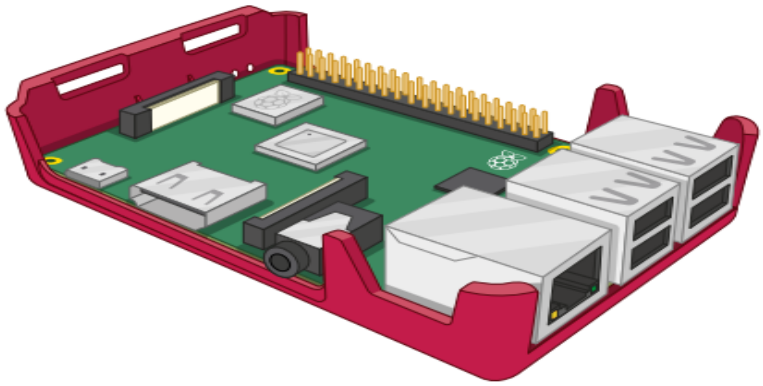
Assembling the case

It should be your first step if you're placing your Pi in a case. If you're using the Official Raspberry Pi Case, start by separating the five pieces: the red base, two white sides, red upper and white lid.

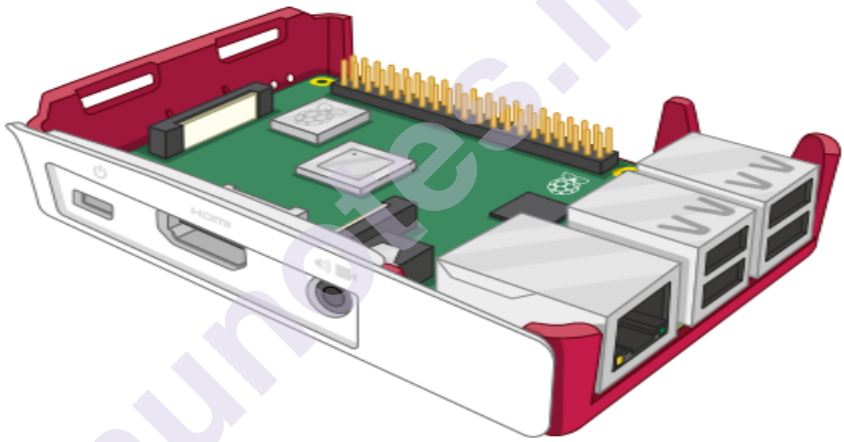
1] Take the base and place it on your left side with the elevated end facing you and the lower end facing you.



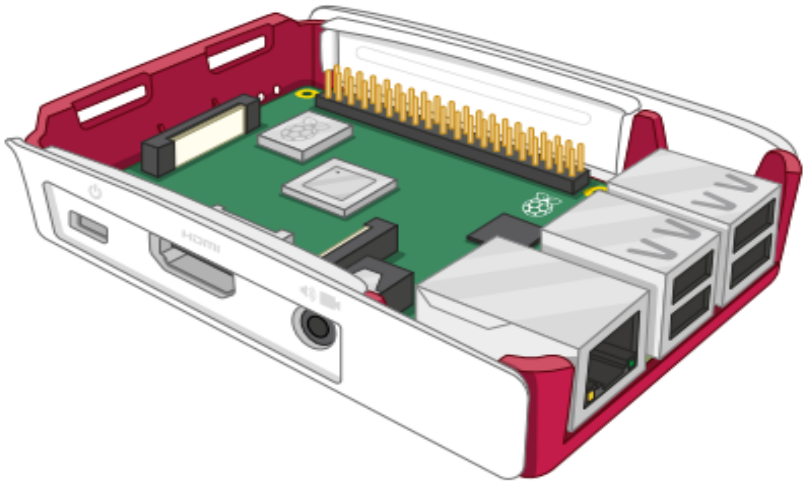
2] Holding the Pi by its USB and Ethernet ports and the GPIO header at the top, insert the left-hand side into the case at an angle, then slowly drop the right-hand side down until it lies flat.



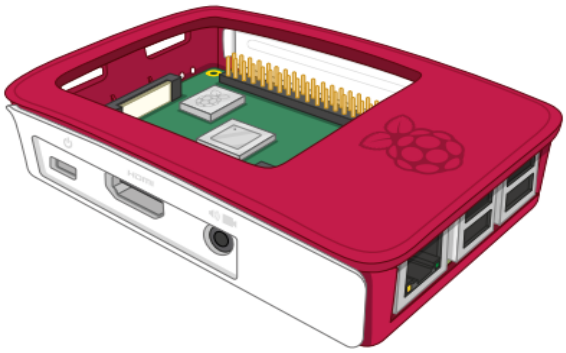
3] Find the one with the cutouts for the power connector, HDMI port, and 3.5 mm AV jack among the two white side parts. Line it up with the Raspberry Pi's ports and carefully press it in until you hear a click.



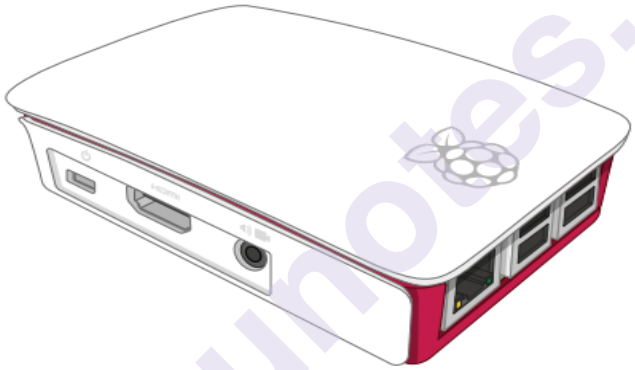
4] Place the solid white side piece on the GPIO header side of the casing and click it in place.



5] Place the two clips on the left of the red plastic upper piece into the matching holes on the left of the base, above the microSD card slot. Push the right-hand side (above the USB ports) down until you hear a click once they're in place.

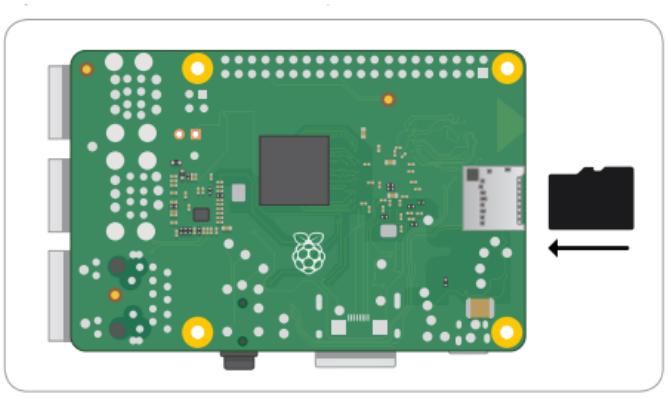


6] Finally, carefully press the white lid down until you hear a click, making that the Raspberry Pi logo is to your right and the small raised clips on its underside are lined up with the hole on the top of the case. Your case is now complete.

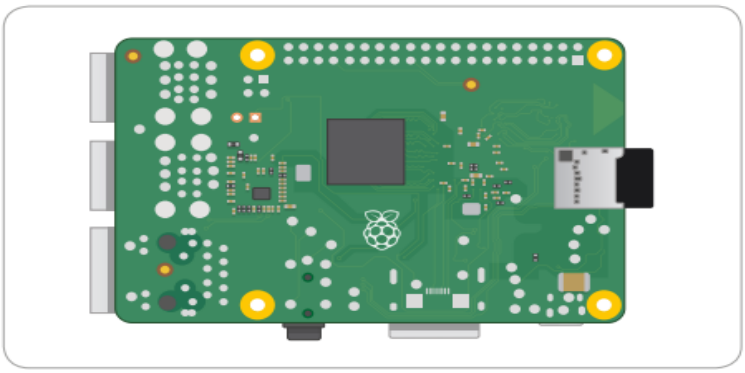


Connecting the microSD card

Turn the Raspberry Pi over and insert the microSD card into the microSD slot with the label facing away from the Pi to install the microSD card, which is the Raspberry Pi's storage. It can only go one way and should go into place without too much difficulty.



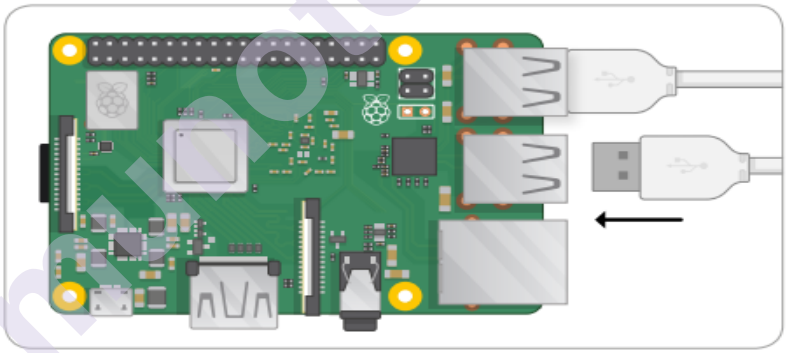
The microSD card will go into the connector and then come to a halt without making a click.



If you want to remove it in the future, simply grab the card's end and carefully pull it out. If you're using an earlier Raspberry Pi, you'll need to gently push the card to unlock it; this isn't necessary if you're using a Raspberry Pi 3 or newer.

Connecting a keyboard and a mouse

Connect the USB connection from the keyboard to one of the Raspberry Pi's four USB ports. Once the keyboard is connected, attach the mouse in the same way.

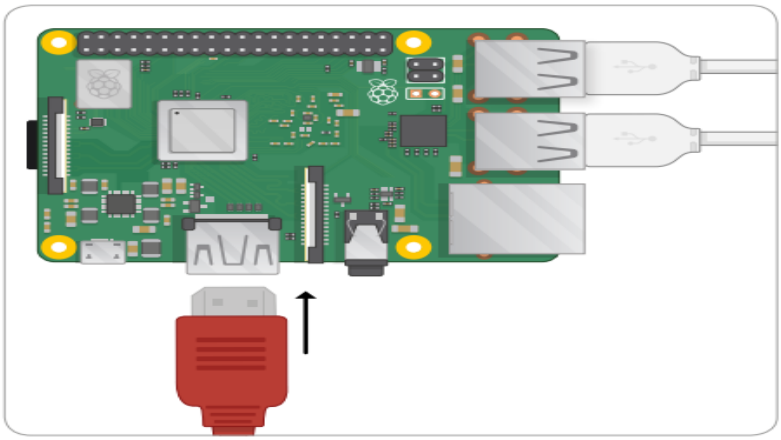


The USB connectors for the keyboard and mouse should slip into place without too much force; if you have to force them in, something is amiss. Make sure the USB connector is pointing in the appropriate direction!

- MOUSE & KEYBOARD: The keyboard and mouse are your primary way of instructing the Raspberry Pi; these are known as input devices in computing, as opposed to the display, which is an output device.

Connecting a display

Connect one end of the HDMI cable to your Raspberry Pi and the other end to your monitor (it doesn't matter which). Look for a port number next to the connector itself if your display has more than one HDMI port; you'll need to switch the TV to this input to see the Pi's display. Don't worry if you can't see a port number: simply switch through each input until you find the Pi.

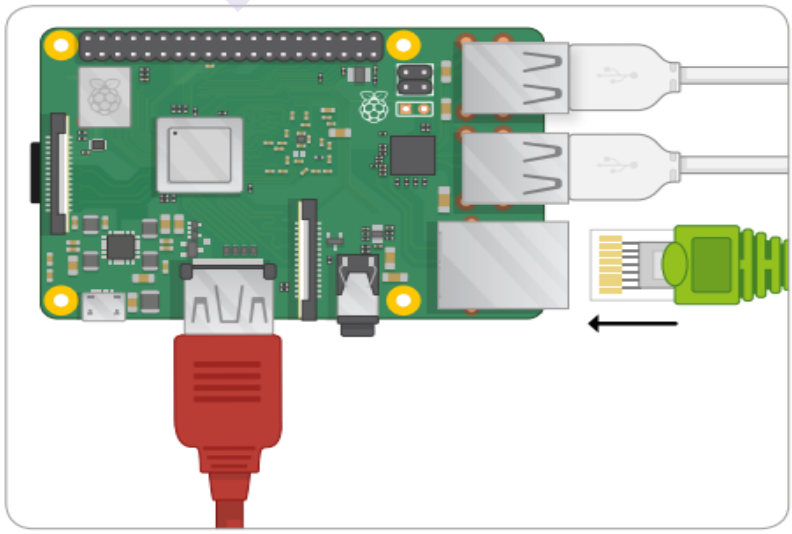


- CONNECTION TO THE TV: It doesn't imply you can't use the Raspberry Pi if your TV or monitor doesn't have an HDMI port. Adapter cables, which can be found at any electronics store, can convert the Raspberry Pi's HDMI port to DVI-D, Display Port, or VGA for use with older computer monitors; they are simply attached to the Pi's HDMI port, and then an appropriate cable is used to connect the adapter cable to the monitor. If your TV only has a composite video or SCART input, you can buy 3.5 mm tip-ring-ring-sleeve (TRRS) adapter cables and composite-to-SCART adapters to plug into the 3.5 mm AV port.

Connecting a network cable (optional)

To connect your Raspberry Pi to a wired network, insert a network cable – also known as an Ethernet cable – into the Ethernet port on the Pi, with the plastic clip facing down, until you hear a click. If the cable needs to be removed, simply squeeze the plastic clip inwards towards the plug and gently slip the cable free.

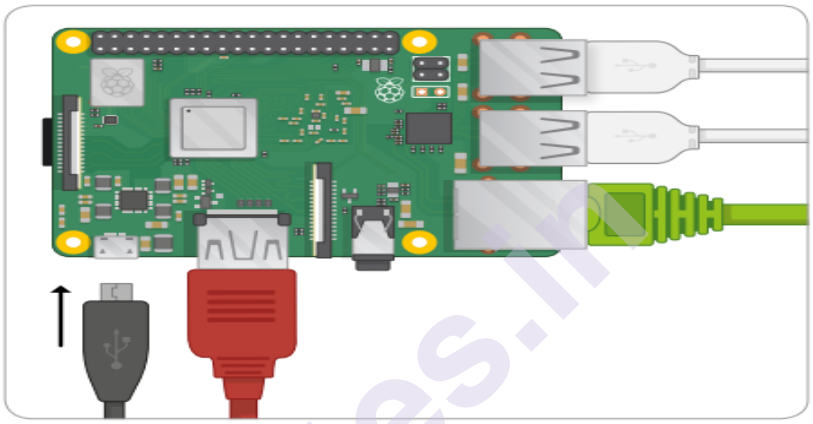
In the same way, attach the opposite end of your network cable to any free port on your network hub, switch, or router.



Connecting a power supply

The last stage in the hardware setup procedure is to connect the Raspberry Pi to a power source, which you should do only when you're ready to set up its software: the Raspberry Pi lacks a power switch and will turn on as soon as it's attached to a live power supply.

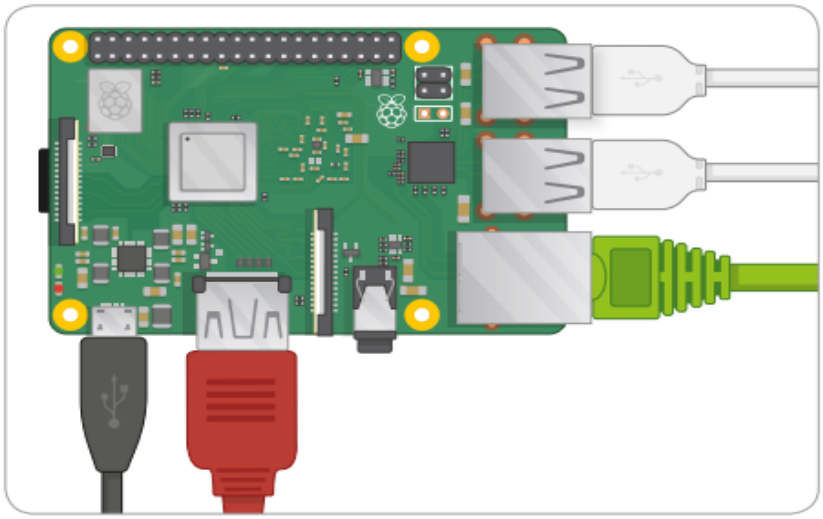
Connect the micro USB end of the power supply cable to the Raspberry Pi's micro USB power connection. It can only travel one way, with the thin part of the connector pointing down, and should softly slide home.



- POWER SUPPLY: If you're using the Official Raspberry Pi Power Supply, you'll see that it comes with multiple mains connectors that are compatible with different nations' sockets. Choose the one that corresponds to the socket type in your nation, and then slide it onto the power supply body until you hear a click.

Finally, connect the power supply to a mains socket and turn it on; the Raspberry Pi will start running instantly.

You've completed the assembly of your Raspberry Pi!



Setting up the software

You'll need to set up the Raspberry Pi's software, particularly its operating system, which regulates what the Pi can do, before you can start using it in earnest. NOOBS, or New Out-Of-Box Software, is designed to make this process as simple as possible by allowing you to choose from a variety of operating systems and have them installed automatically. Even better, you can accomplish all of this with only a few mouse clicks.

You'll see a screen with the Raspberry Pi logo on it and a small progress window at the upper-left when the Pi is initially switched on, or booted, with a fresh installation of NOOBS on its microSD card. You'll see the screen shown in Figure 1.40 after a brief wait, which might take up to a minute the first time you use the NOOBS microSD card.

Figure 1.40 The NOOBS menu without any operating system installed

- ARE THERE NO PICTURES?

  Check that you're using the correct input if you can't see the Raspberry Pi on your screen. If your TV or monitor has multiple HDMI inputs, use the ‘Source' or ‘Input' buttons to cycle through each one until you get the NOOBS menu.

  This is the NOOBS menu, which allows you to select an operating system for your Raspberry Pi. Raspbian, a version of the Debian Linux operating system customised exclusively for the Raspberry Pi, and LibreELEC, a version of the Kodi Entertainment Centre software, are supplied as standard with NOOBS. You can also download and

install different operating systems if the Pi is connected to the network - either through a wired connection or using the ‘Wifi networks (w)' option from the top bar of icons.

Use the mouse to draw a cross in the box to the left of Raspbian Full: position the pointer at the white box and click once with the left mouse button to begin installing an operating system. When you've done so, the ‘Install I menu icon will no longer be greyed-out, indicating that your operating system is ready to install (Figure 1.41)

Figure 1.41 Choosing an operating system to install through NOOBS

When you press the left mouse button on the ‘Install (i)' icon, a warning notice appears, informing you that installing the operating system would overwrite any data currently saved on the microSD card, except for NOOBS, which will remain intact. The installation process will begin once you click ‘Yes' (Figure 1.42)

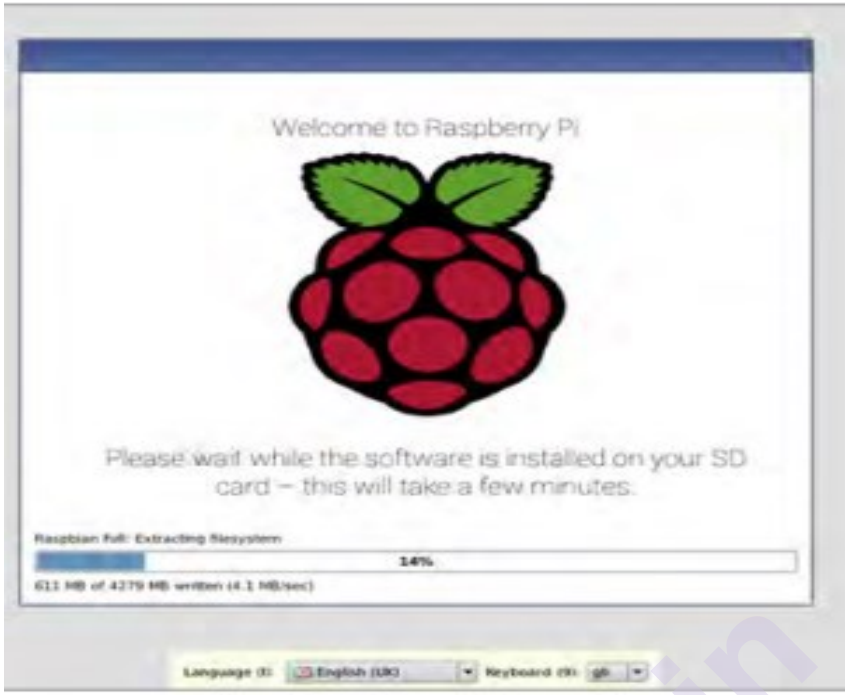


Figure 1.42 Installing the Raspbian operating system

Depending on the speed of your microSD card, the installation process can take anywhere from 10 to 30 minutes. Progress is presented in a bar down the bottom of the window as the operating system is installed, and you'll watch a slide show outlining some of its important features.

- **WARNING!**

It's critical that the installation isn't interrupted because doing so risks destroying the software through a process known as data corruption. While the operating system is being installed, do not remove the microSD card or unplug the power cable; if something happens to interrupt the installation, unplug the Raspberry Pi from its power supply, then press the SHIFT key on the keyboard while reconnecting the Raspberry Pi to its power supply to bring up the NOOBS menu. This is known as recovery mode, and it's a terrific way to get a Pi back into working condition when its software has been corrupted. After a successful installation, it also allows you to access the NOOBS menu, where you can reinstall the operating system or install one of the other operating systems.

When the installation is complete, a popup with a ‘OK' button will appear; click this to restart the Pi in its newly installed operating system. The boot messages will scroll up the screen (Figure 1.43), and the first time you boot into Raspbian, it may take a minute or two as it adapts to make the greatest use of the free space on your microSD card. Things will move more rapidly the next time you boot.

Figure 1.43 The Raspbian boot messages

Finally, before the Raspbian desktop and setup wizard display, you'll see a window with the Raspberry Pi logo on it, as shown in Figure 1.44. Your operating system has now completed installation and is ready for configuration.

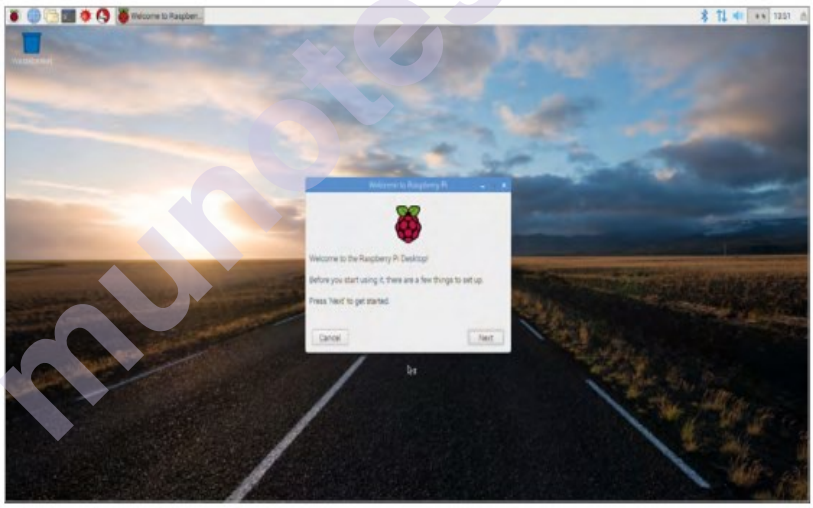


Figure 1.44 The Raspbian desktop

1.6 RASPBERRY PI BOOT

There is no BIOS on the Raspberry Pi. Your SDCard contains GPU firmware that allows you to use the GPU. The GPU kicks off the ARM processor and loads the Linux kernel. Hundreds of documents detailing how that procedure works can be found on the internet.

On the Raspberry Pi, Android is a NON-STARTER.

Windows 10 IoT isn't the same as the Windows you're used to (and despise) on your laptop. It's a specialized Windows internet of things system that only operates on an RPI2.

1.6.1 Learn how this small SoC boots without BIOS

Instead of BIOS, the Raspberry Pi uses "firmware." To add to the confusion, all B models require this firmware to be installed on the SD card. You won't even get error messages if your SD card isn't working or if you neglect to put the firmware on it. The Raspberry Pi will do nothing. The simplest method for dual booting a Pi is to use different SD cards. The SD card functions similarly to a hard disc on a desktop or laptop computer; swapping it, however, allows you to use a different operating system. It's similar to a Gameboy cartridge. The GPU handles everything, after which the kernel is loaded and the CPU is turned on.

1.6.2 Configuring boot sequences and hardware

To begin, you must understand that the Raspberry Pi does not operate in the same way as a traditional desktop computer. The graphics processor starts up before the ARM processor!

Here's a diagram of the Raspberry Pi with the Broadcom BCM2835 SoC highlighted before we dive into the details.

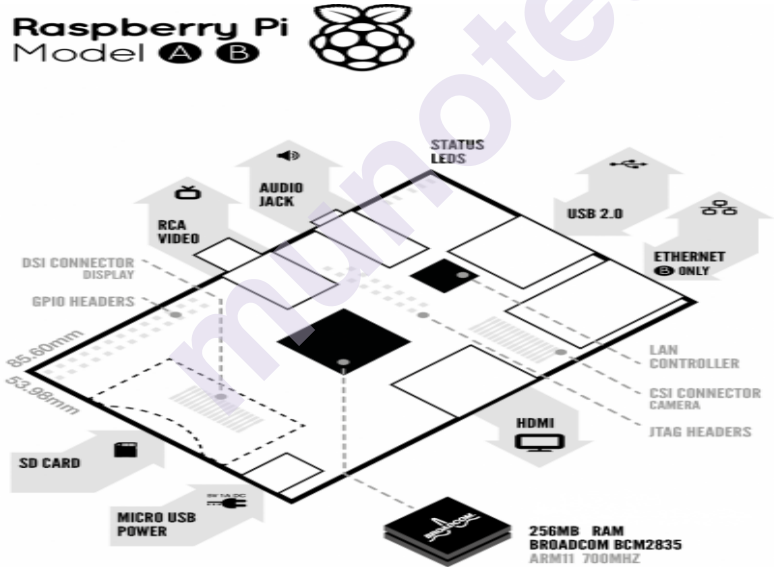


Figure 1.45 Raspberry Pi with the Broadcom BCM2835 SoC

The ARM CPU, VideoCore Graphics Processor, ROM (Read-Only Memory) chips, SDRAM, and other components are all found on the SoC (or System-on-Chip). Consider a SoC to be a combination of your motherboard and CPU crammed into a single chip.

The initial bits of code that run when you turn on your Raspberry Pi are saved in a ROM chip in the SoC that was integrated into the Pi during manufacturing! The first-stage bootloader is what it's called. On startup, the SoC is hardwired to run this code on a tiny RISC Core (Reduced Instruction

Set Computer). It's utilized to access the second-stage bootloader by mounting the FAT32 boot partition on your SDCard. So, what exactly is this SD Card's "second-stage bootloader"? It's called 'bootcode.bin'. If you had mounted the SD Card in Windows, you might have seen this file. Here's when it gets tricky. Your ARM CPU (which is in reset) and RAM have not yet been initialized by the first-stage bootloader. As a result, the second-stage bootloader must operate on the GPU as well. The bootloader.bin file is loaded and executed from the GPU's 128K 4 way set associative L2 cache. This activates the RAM and loads start.elf from your SD Card. This is the most significant of the three-stage bootloaders. It's the GPU's firmware, which means it has the settings or, in our case, instructions for loading the settings from config.txt on the SD Card. The config.txt file can be thought of as the 'BIOS settings' (as is mentioned in the forum). The following are some of the options available to you:

- **arm_freq**: It is the ARM frequency in MHz. Default frequency is 700MHz.
- **gpu_freq**: This sets core_freq, h264_freq, isp_freq, v3d_freq together.
- **core_freq**: It is the GPU processor core frequency and is expressed in MHz. Default frequency is 250MHz.
- **h264_freq**: This is the hardware video block frequency in MHz. Default is 250MHz.
- **isp_freq:** It illustrates the image sensor pipeline block frequency in MHz. Default is 250MHz.
- **v3d_freq:** It describes the frequency of 3D block in MHz. Default is 250MHz.
- **sdram_freq:** This is the SDRAM frequency expressed in MHz. Default is 400MHz.

In addition, the start.elf divides the RAM between your GPU and the ARM CPU. Only the address space left over from the GPU address space is accessible to the ARM. The MMU (Memory Management Unit) of the VideoCore maps the physical addresses detected by the ARM core to another address in the VideoCore (0xC0000000 and beyond). Because the config.txt is loaded after the split, you can't specify the dividing amounts there. However, the SD Card has a variety of .elf files with various splits. You can rename those files to start.elf and boot the Pi, according on your needs. The GPU wins every time on the Raspberry Pi!

The start.elf additionally loads cmdline.txt if it exists, in addition to loading config.txt and splitting RAM. It specifies the command line parameters for the kernel to be loaded. The boot process has now reached its end. The commencement .elf loads kernel.img, the binary file containing the OS kernel (DUH!?) and relieves the CPU reset. The ARM CPU then executes the kernel.img instructions, which loads the operating system.

Here's how it goes:

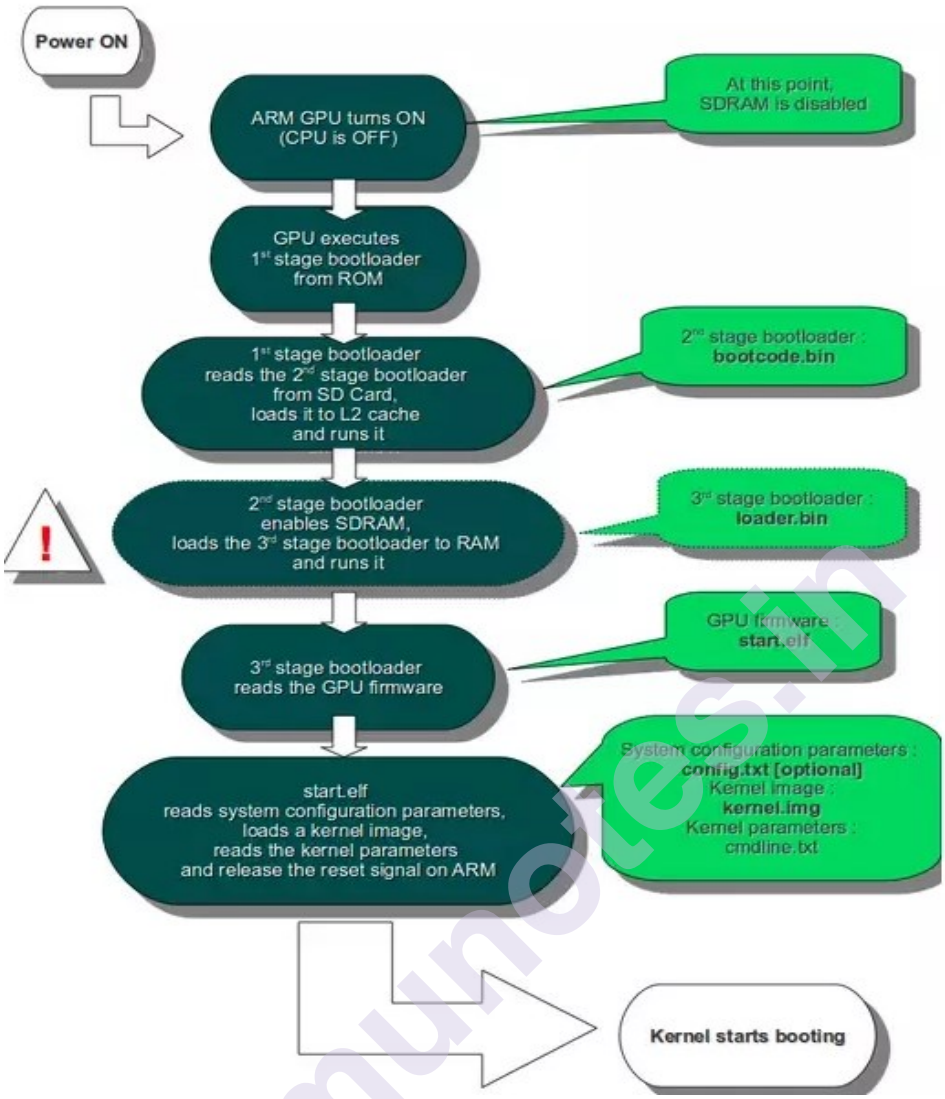


Figure 1.46 Flow process of boot sequence configuration

So the Raspberry Pi, unlike a PC, does not require a BIOS sequence because the required startup functions are incorporated into the GPU.

1.7 SUMMARY

In this chapter we presented the concept and approaches used in creating a system-on-chip (SoC) based on a microprocessor core, as well as the microprocessor core itself, were introduced in this course. The reader gained a better grasp of how SoCs are built and used, as well as why current processors are designed the way they are.

The ARM images bring reality to topics that can otherwise appear ethereal to the reader who only wants to know the basic principles; the general principles reveal the logic for the ARM being as it is to the reader who wants to understand the design of the ARM.

The technical insights about the Raspberry Pi are discussed that covered up the basic introduction of Pi, various generations of Raspberry Pi models. The content acquainted the reader to set up their own operating system and can also connect the wiring and circuits directly with the Raspberry Pi board along with the discussion of onboard hardware components. Lastly, the unit concluded with the discussion on configuring the boot sequence and hardware.

1.8 LIST OF REFERENCES

1) Learning Internet of Things, Peter Waher, Packt Publishing(2015)

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

3) Abusing the Internet of Things, Nitesh Dhanjani, O'Reilly

4) Michael J. Flynn, Wayne Luk, Computer System Design: System on Chip, John Wiley and Sons Inc. 2011, ISBN 978-0-470-64336-5

5) SystemC: From the Ground Up, 2nd Edition, D.C. Black, J Donovan, B. Bunton, A. Keist, Springer 2010, ISBN 978-0-387-69958-5.

6) On-Chip Communication Architectures, System on Chip Interconnect, S. Pascricha and N. Dutt, Morgan Kaufmann-Elsevier Publishers 2008, ISBN 978-0-12-373892-9.

7) https://www.cs.cmu.edu/afs/cs/academic/class/15462-f11/www/lec_slides/lec19.pdf

8) https://www.researchgate.net/publication/260687001_GPU_computing

9) https://cdn.iiit.ac.in/cdn/cstar.iiit.ac.in/~kkishore/GPUArchitecture.pdf

10) http://web.eecs.umich.edu/~prabal/teaching/eecs373f12/readings/ARM_Architecture_Overview.pdf

11) https://web.sonoma.edu/users/f/farahman/sonoma/courses/es310/310_arm/lectures/Chapter_3-and-1_ARM.pdf

12) https://www.cs.unca.edu/~bruce/Fall14/360/RPi UsersGuide.pdf

13) http://meseec.ce.rit.edu/551-projects/spring2017/2-3.pdf

1.9 UNIT END EXERCISES

1) Explain the concept of SoC

2) Write a note on significance and design challenges of SoC.

3) Describe the advantages, disadvantages and applications of system on chip.

4) Write a short note on system on chip.

5) Explain the architecture of FPGA.
6) Write a note on working and applications of FPGA.
7) Describe the architecture and applications of GPU.
8) What is the difference between GPU and CPU?
9) Write a note on APU.
10) Explain the configuration of compute units.
11) Describe ARM8 architecture.
12) Write a note on Raspberry Pi.
13) Explain the various generations of Raspberry Pi.
14) Describe the Raspberry Pi operating system.
15) Describe the hardware of Raspberry Pi.
16) Write a note on configuring boot sequence and hardware along with its flow diagram.

❄❄❄❄❄❄❄

2

PROGRAMMING RASPBERRY PI

Unit Structure

2.0 OBJECTIVES

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

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

2.1 INTRODUCTION

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

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

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

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

2.3 RASPBERRY PI AND LINUX

2.3.1 ABOUT RASPBIAN

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

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

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

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

2.3.1.1 HISTORY OF RASPBIAN

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

- **Raspbian Wheezy**

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

- **Raspbian Jessie**

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

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

would appear on the screen. Under voltage was indicated by a lightning bolt, and temperature warnings were indicated by a thermometer.

- **Raspbian Stretch**

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

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

- **Raspbian Buster**

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

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

2.3.1.2 FEATURES OF RASPBIAN

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

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

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

Basic features are as follows:

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

2.3.1.3 WHO SHOULD USE THE RASPBERRY PI OPERATING SYSTEM?

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

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

2.4 LINUX COMMANDS

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

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

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

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

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

- **Linux directory commands**

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

Syntax: pwd

Output:

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

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

Syntax: mkdir <directory name>

Output:

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

3] rmdir Command: used for deleting a directory.

Syntax: rmdir <directory name>

Output:

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

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

Syntax: ls

Output:

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

5] cd Command: for changing the current directory.

Syntax: cd <directory name>

Output:

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

- **Linux File commands**

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

Syntax:

touch <file name>

touch <file1> <file2> ....

Output:

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

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

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

To create a file, execute it as follows:

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

// Enter file content

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

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

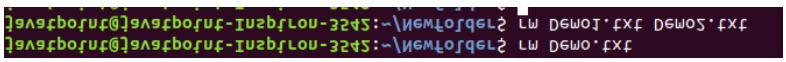
Output:

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

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

Syntax: rm <file name>

Output:

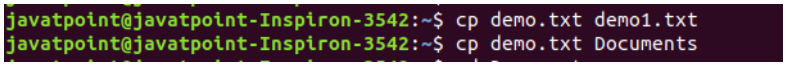


9] **cp command: It is used for copying a file or directory.**

Syntax: To copy in the same directory:

cp <existing file name> <new file name>

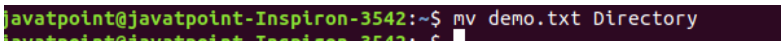
Output:



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

Syntax: mv <file name> <directory path>

Output:



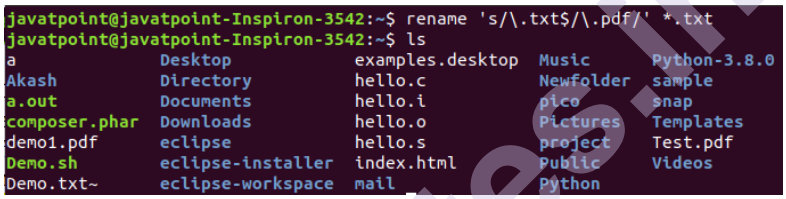
11] rename Command: This command is used to rename the large group of files

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

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

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

Output:

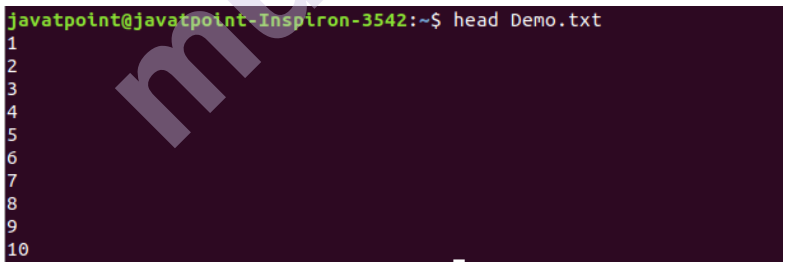


- **Linux File Content Commands**

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

Syntax: head <file name>

Output:



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

Syntax: tail <file name>

Output:

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

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

Syntax: tac <file name>

Output:

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

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

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

ENTER key: To scroll down page by line.

Space bar: To advance to the next page.

b key: To return to the previous page.

/ key: To search the string.

Syntax: more <file name>

Output:

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

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

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

(require 'python)
(require 'cl)

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

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

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

Syntax: less <file name>

Output:

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

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

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

(require 'python)
(require 'cl)

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

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

- **Linux User Commands**

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

Syntax: su <user name>

Output:

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

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

Syntax: id

Output:

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

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

Syntax: useradd username

Output:

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

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

Syntax: passwd <username>

Output:

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

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

Syntax: groupadd <group name>

Output:

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

- **Linux Filter Commands**

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

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

Output:

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

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

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

Output:

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

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

Syntax: command | grep <searchWord>

Output:

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

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

Syntax: comm <file1> <file2>

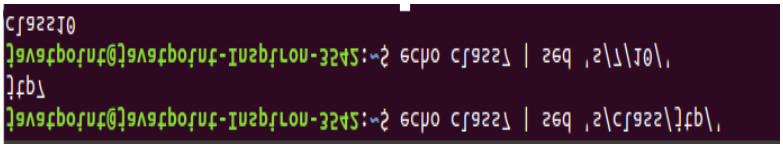
Output:

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

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

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

Output:



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

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

Output:

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

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

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

Output:

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

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

Syntax: command <fileName> | uniq

Output:

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

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

Syntax: wc <file name>

Output:

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

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

Syntax:

od -b <fileName> // Octal format

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

od -c <fileName> // ASCII character format

Output:

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

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

Syntax: sort <file name>

Output:

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

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

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

Output:

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

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

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

Output:

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

- **Linux Utility Commands**

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

Following the find command, the following symbols are used:

(.) For the current directory

(/): for the root

Syntax: find . -name "*.pdf"

Output:

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

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

Syntax: locate <file name>

Output:

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

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

Syntax: date

Output:

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

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

Syntax: cal<

Output:

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

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

Syntax: sleep <time>

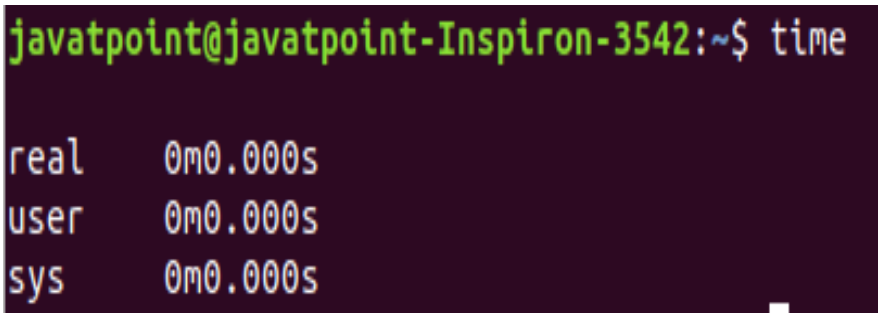
Output:

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

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

Syntax: time

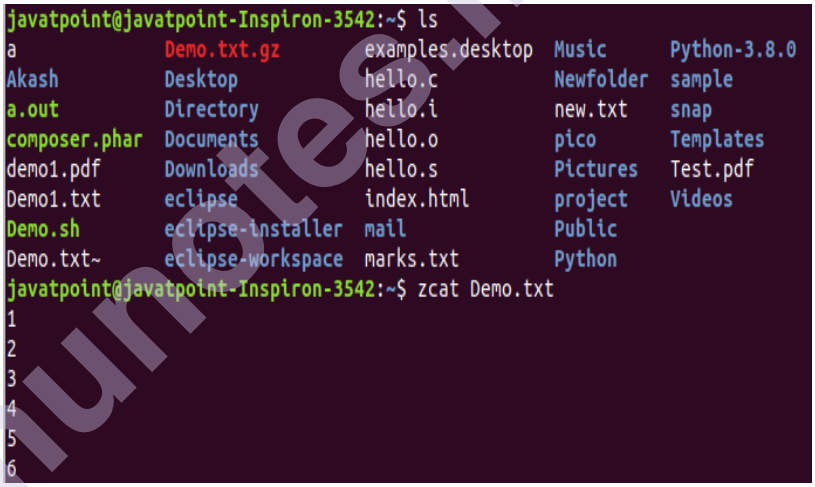
Output:



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

Syntax: zcat <file name>

Output:



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

Syntax: df

Output:

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

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

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

Output:

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

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

Syntax: exit

Output:

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

It will exit the terminal after pressing the ENTER key.

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

Syntax: clear

Output:

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

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

- **Linux Networking Commands**

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

Syntax: ip a or ip addr

Output:

```
javatpoint@javatpoint-Inspiron-3542:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp7s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOW
N group default qlen 1000
    link/ether 74:e6:e2:02:93:b8 brd ff:ff:ff:ff:ff:ff
3: wlp6s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP gro
up default qlen 1000
    link/ether 00:71:cc:00:e2:89 brd ff:ff:ff:ff:ff:ff
    inet 192.168.43.240/24 brd 192.168.43.255 scope global dynamic noprefixroute
 wlp6s0
       valid_lft 2296sec preferred_lft 2296sec
    inet6 fe80::8c59:e84e:1670:27cc/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

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

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

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

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

Output:

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

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

Syntax: ping <destination>

Output:

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

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

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

Output:

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

2.5 CONFIGURING RASPBERRY PI WITH LINUX COMMANDS

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

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

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

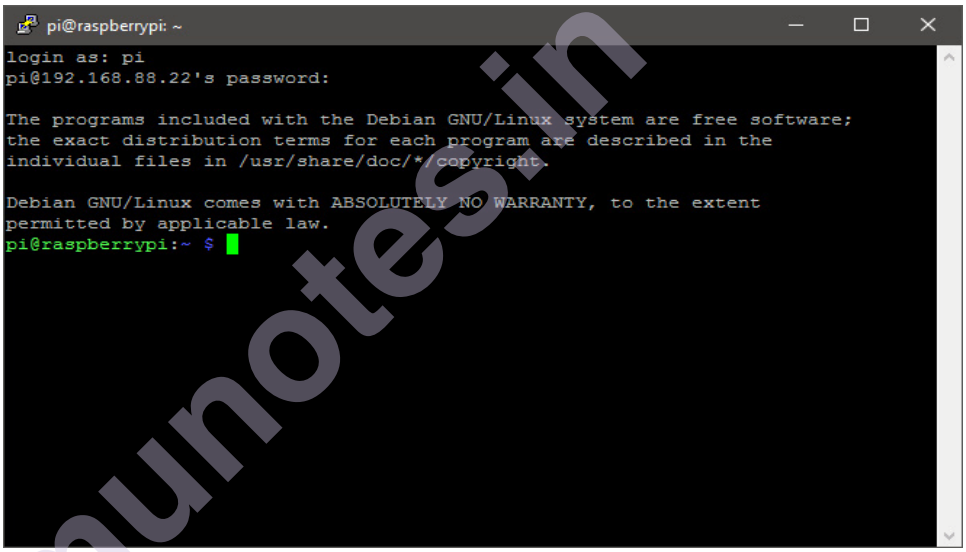
  When utilizing a command-line, you generally instruct the Raspberry Pi to complete tasks by typing instructions into the terminal, as opposed to the traditional method of interaction, which involves using a mouse. You might assume it's easier to communicate with a Raspberry Pi by connecting a display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

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

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

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

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



- **Updating the system**

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

sudo apt-get update

sudo apt-get upgrade

sudo apt-get dist-upgrade

sudo rpi-update

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

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

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

- **Navigating through files and folders**

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

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

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

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

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

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

- **Performing file and folder operations**

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

  mkdir: for creating a new directory

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

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

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

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

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

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

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

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

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

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

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

  nano newproject.txt

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

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

- **Raspberry Pi hardware information**

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

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

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

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

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

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

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

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

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

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

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

  htop

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

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

  ifconfig

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

- **Shutdown and restart your Raspberry Pi**

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

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

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

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

2.6 SUMMARY

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

2.7 LIST OF REFERENCES

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

UNIT END EXERCISES

1] Write a note on Raspbian.

2] Explain the history and features of Raspbian.

3] Write a detailed note on different Linux commands.

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

❄❄❄❄❄❄❄

3

PROGRAMING INTERFACES

Unit Structure

3.0 OBJECTIVES

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

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

3.1 INTRODUCTION TO NODE.JS

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

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

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

3.1.1 WHY SHOULD YOU USE NODE.JS?

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

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

3.1.2 FEATURES OF NODE.JS

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

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

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

- The code line above examines reading a file named Sample.txt. In other programming languages, the next line of processing would take place only after the full file has been read.
- However, in the case of Node.js, the definition of the function (‘function(error,data)') is the most significant part of the code to pay attention to. A callback function is what this is called.

- So, in this case, the file reading activity will begin in the background. While the file is being read, other processing can take place at the same time. This anonymous function will be called whenever the file read process is complete, and the text "Reading Data done" will be written to the console log.

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

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

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

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

3.1.3 WHO MAKES USE OF NODE.JS?

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

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

3.1.4 WHEN SHOULD YOU USE NODE.JS?

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

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

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

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

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

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

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

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

3.1.5 WHEN WILL YOU AVOID USING NODE.JS?

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

3.1.6 COMPONENTS OF NODE.JS

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

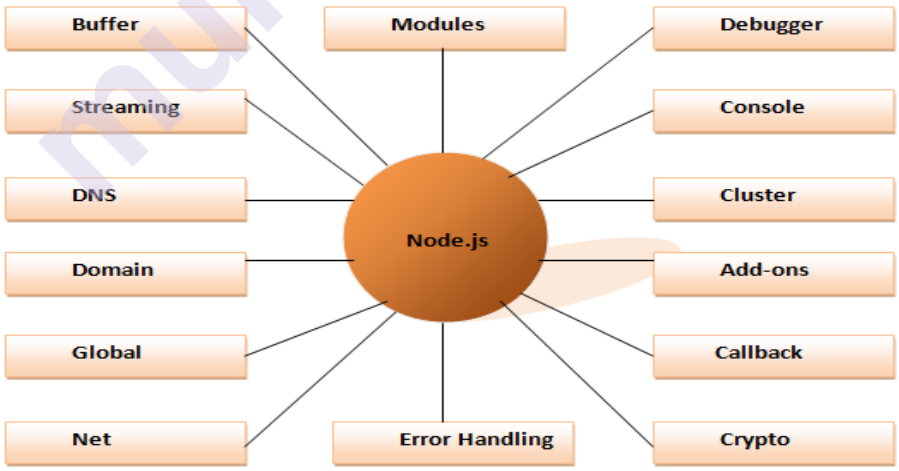


Figure 1.1 Components of Node.js

3.1.7 NODE.JS FRAMEWORKS AND TOOLS

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

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

- **Nx:** It is a full-stack monorepo development toolkit that includes NestJS, Express, React, Angular, and more. Nx enables you to scale your development from a single team producing a single app to several teams working on multiple apps!

- **Sapper:** Sapper is a web application framework with a beautiful development experience and configurable filesystem-based routing for web applications of all sizes. Offers SSR as well as other services!

- **Socket.io:** It is a network application development platform that uses real-time communication.

- **Strapi:** Strapi is an open-source Headless CMS that allows developers to use their preferred tools and frameworks while also allowing editors to manage and distribute their content simply. Strapi helps the world's largest enterprises to expedite content delivery while creating stunning digital experiences by making the admin panel and API expandable through a plugin system.

3.2 PYTHON

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

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

3.2.1 PYTHON 2 Vs. PYTHON 3

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

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

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

in Python 3 that can be used as print("something") to print something to the console.

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

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

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

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

3.2.2 HISTORY OF PYTHON

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

3.2.3 WHY TO LEARN PYTHON?

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

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

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

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

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

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

3.2.4 CHARACTERISTICS OF PYTHON

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

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

3.2.5 APPLICATIONS OF PYTHON

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

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

3.3 SUMMARY

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

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

3.4 LIST OF REFERENCES

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

3.5 UNIT END EXERCISES

1] Write a short note on Node.js.

2] Discuss the concept of Python.

❄❄❄❄❄❄❄

4

RASPBERRY PI INTERFACES

Unit Structure

4.0 OBJECTIVES

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

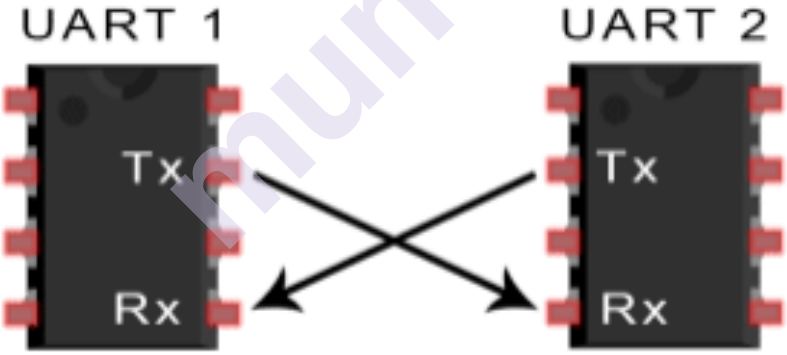
- Understand the concept and applications of communication interfaces
- Introduce with various raspberry pi communication interfaces such as UART, GPIO, I2C, SPI along with its characteristics, working and its applications point of view

4.1 UART

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

4.1.1 INTRODUCTION TO UART COMMUNICATION

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



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

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

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

4.1.2 WHY UART IS USED?

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

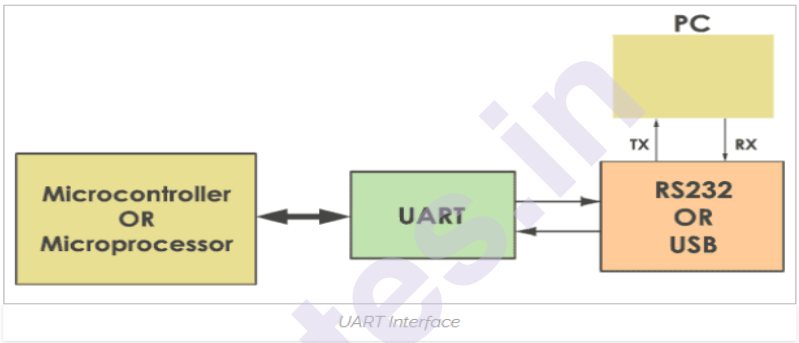


Figure 4.2 UART interface

4.1.3 BLOCK DIAGRAM

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

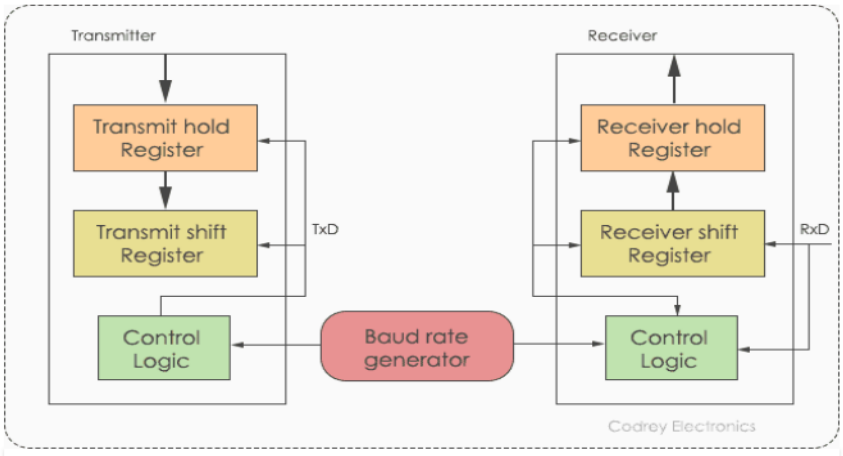


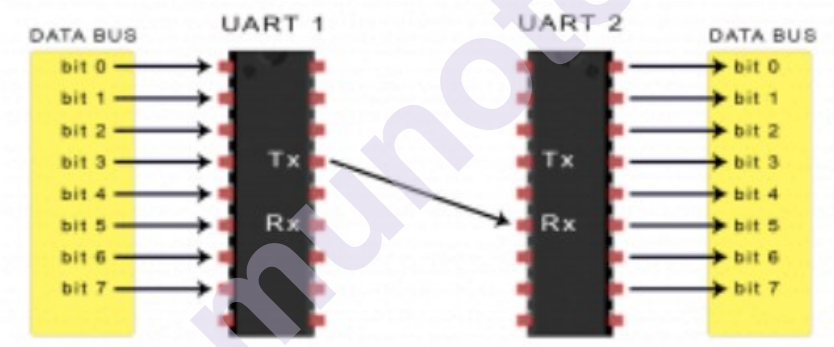
Figure 4.3 UART block diagram

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

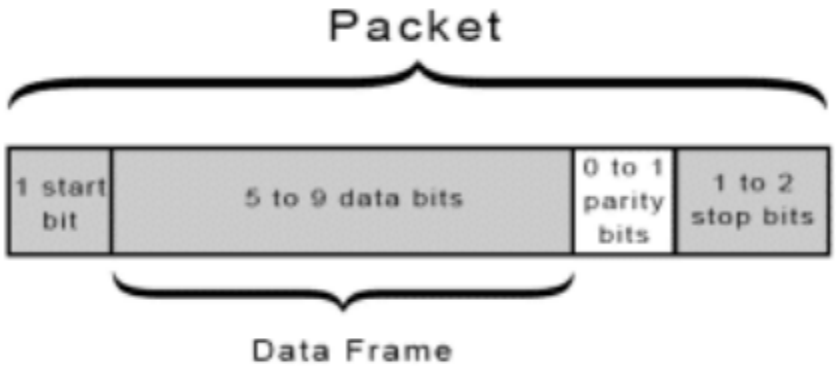
A read or write control logic is also provided to determine when to read or write. The baud rate generator can produce speeds ranging from 110 bps to 230400 bps. For faster data transfer, microcontrollers typically use higher baud rates such as 115200 and 57600. Slower baud rates of 4800 and 9600 are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

4.1.4 HOW UART WORKS

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



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



Start Bit

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

Data Frame

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

Parity

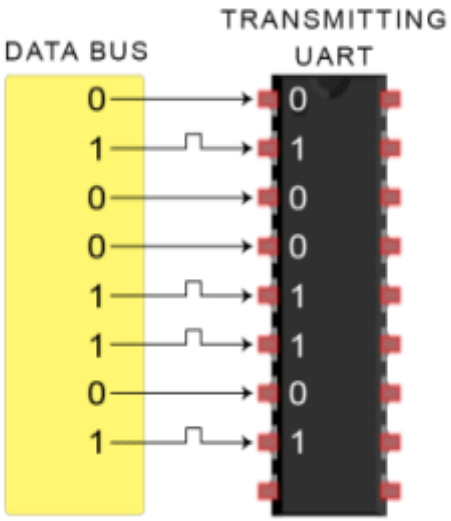
The evenness or oddness of a number is described by parity. The receiving UART uses the parity bit to determine if any data has changed during transmission. Electromagnetic radiation, mismatched baud rates, and long-distance data transmissions can all alter bits. After reading the data frame, the receiving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

Stop bit

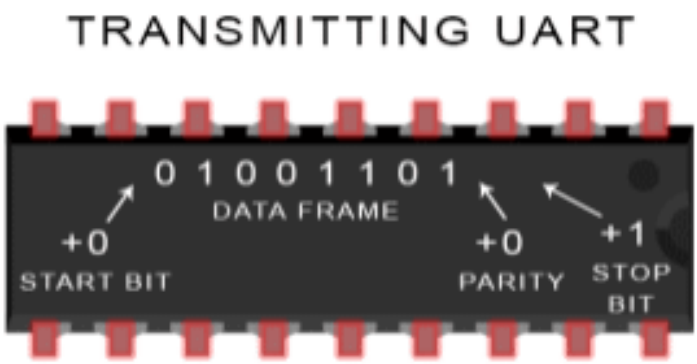
The sending UART drives the data transmission line from a low voltage to a high voltage for at least two bit lengths to signify the end of the data packet.

4.1.5 STEPS OF UART TRANSMISSION

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

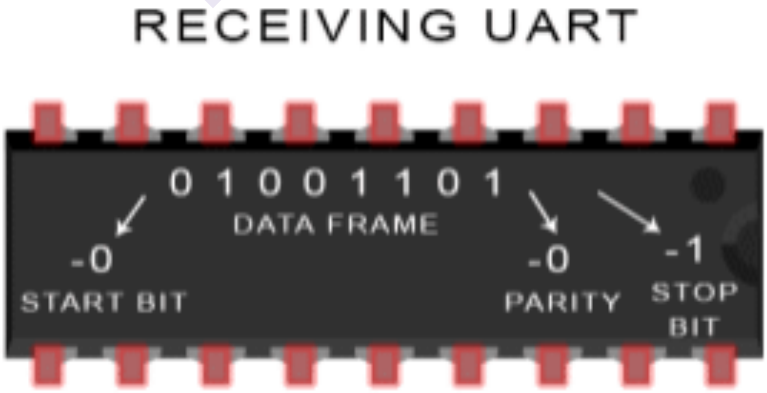


2. The starting bit, parity bit, and stop bit(s) are added to the data frame by the transmitting UART.

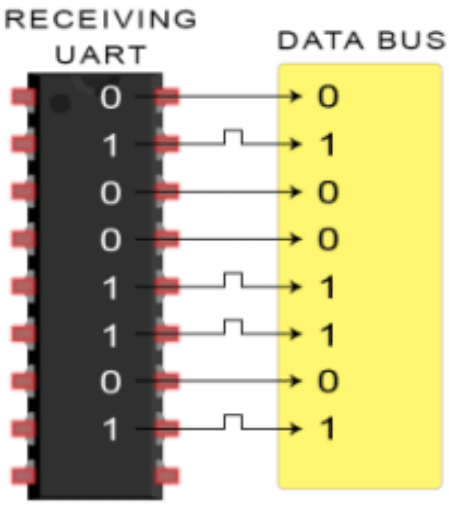


3. From the transmitting UART to the receiving UART, the full packet is transferred serially. The data line is sampled by the receiving UART at the specified baud rate.

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



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



4.1.6 ADVANTAGES OF UART

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

4.1.7 DISADVANTAGES OF UART

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

4.2 GPIO

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

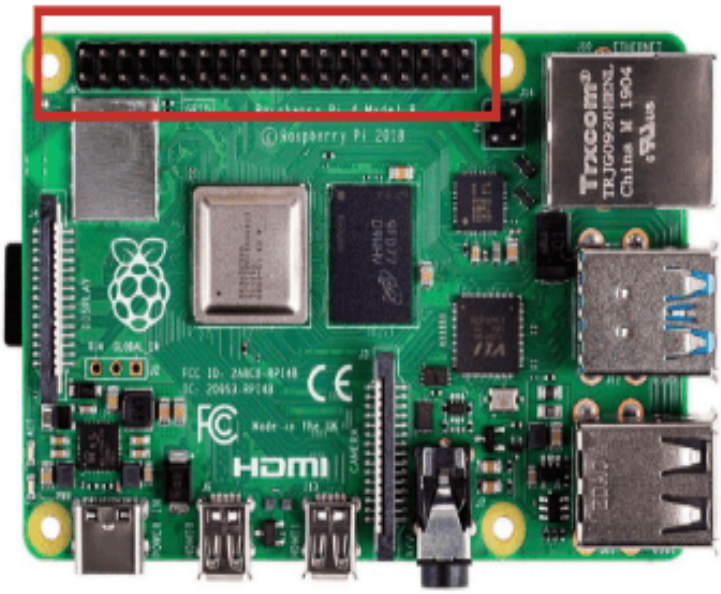


Figure 1.4 Raspberry Pi4 40 Pin GPIO Header

4.2.1 PURPOSE OF THE PERIPHERAL

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

4.2.2 FEATURES

The following are the characteristics of the GPIO peripheral.

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

4.2.3 FUNCTIONAL BLOCK DIAGRAM

Figure 1.5 below represents the GPIO peripheral block diagram

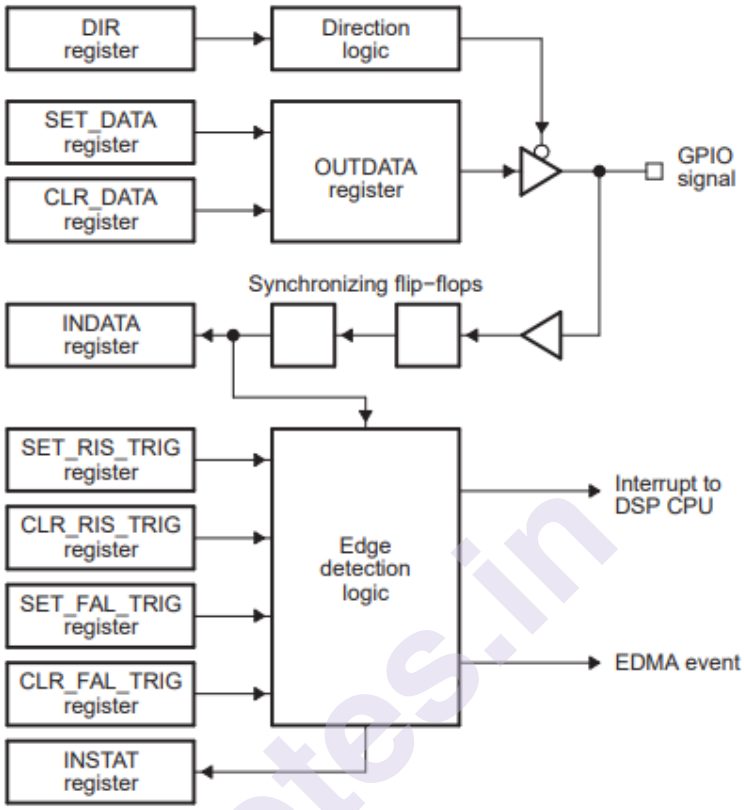


Figure 1.5 GPIO peripheral block diagram

4.2.4 RASPBERRY PI GPIO PINOUT

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

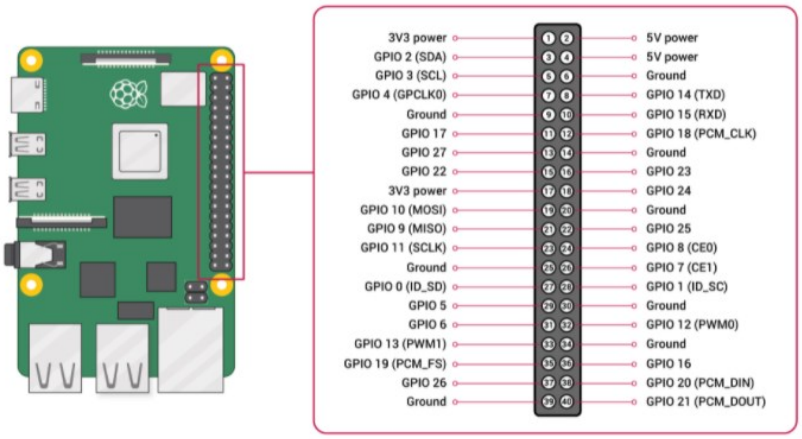


Figure 1.6 Raspberry Pi4 GPIO pin header

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

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

4.2.5 CONFIGURING GPIO PIN

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

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

sudo apt-get update

sudo apt-get upgrade

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

sudo apt-get install rpi.gpio

4.2.6 ESSENTIAL PRODUCTS FOR RASPBERRY PI GPIO

- Grove Base Hat for Raspberry Pi

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

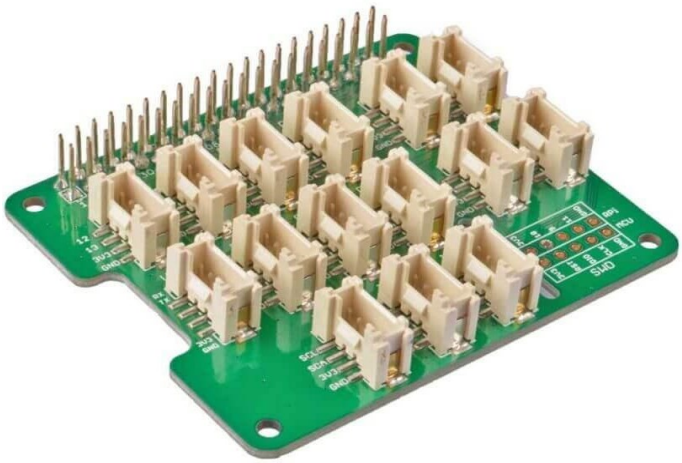
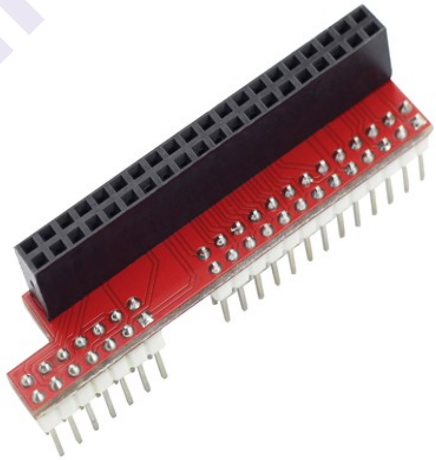


Figure 1.7 Grove Base Hat for Raspberry Pi

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

- Raspberry Pi 40pin to 26pin GPIO Board

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



4.3 I2C

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

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

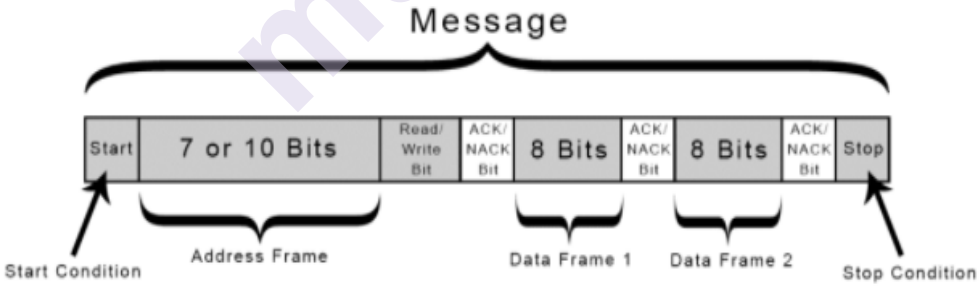
- SDA (Serial Data): The data transmission and reception line between the master and slave.
- Serial Clock Line (SCL): This is the line that carries the clock signal.

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

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

4.3.1 WORKING OF I2C

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



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

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

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

- **Addressing**

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

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

- **Read/Write bit**

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

- **Data Frame**

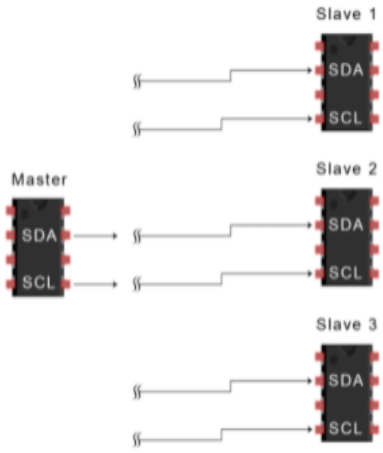
  The initial data frame is ready to be delivered after the master detects the ACK signal from the slave.

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

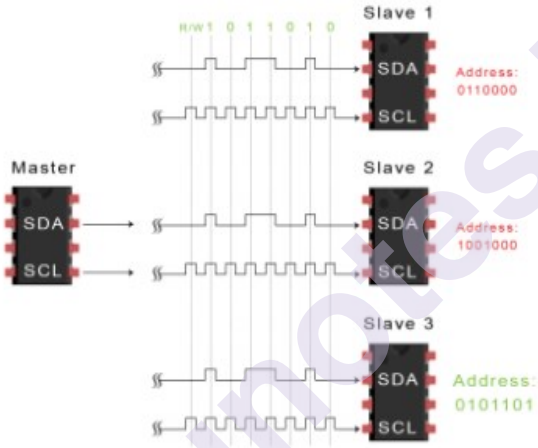
  The master can send a stop condition to the slave to interrupt the transmission when all of the data frames have been sent. After a low to high transition on the SCL line, the stop condition is a voltage transfer from low to high on the SDA line, with the SCL line remaining high.

4.3.2 I2C DATA TRANSMISSION STEPS

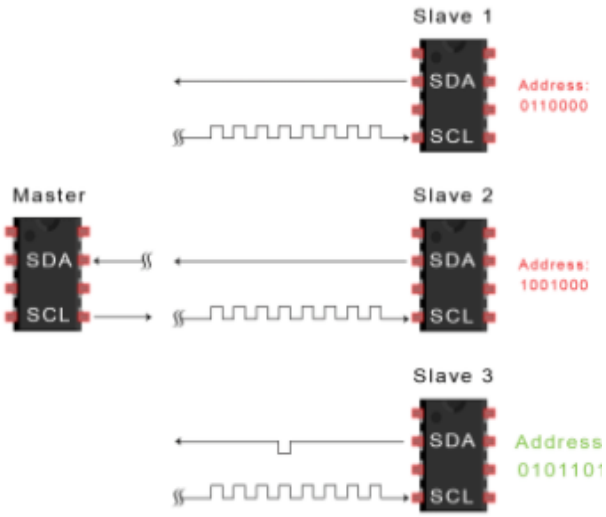
1. Before moving the SCL line from high to low, the master sends the start condition to all linked slaves by switching the SDA line from high to low voltage:



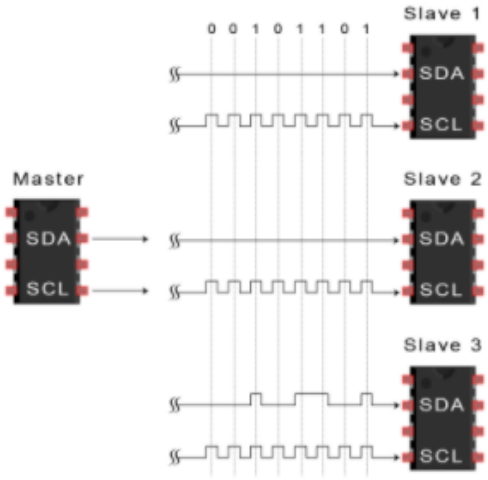
2. The master sends the read/write bit and the 7 or 10 bit address of the slave it wants to connect with to each slave:



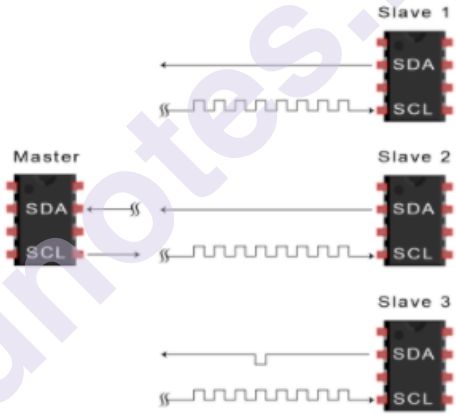
3. Each slave checks the address sent by the master against its own. The slave returns an ACK signal by pulling the SDA line low for one bit if the addresses match. The slave leaves the SDA line high if the master's address does not match the slave's own address.



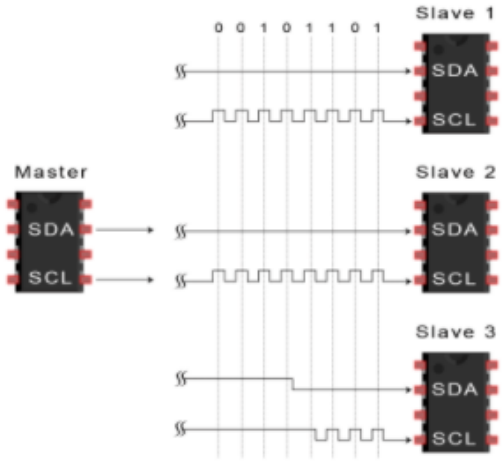
4. The data frame is sent or received by the master:



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

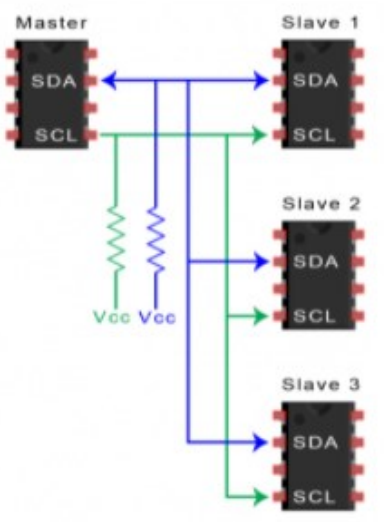


6. The master transmits a stop condition to the slave by switching SCL high before switching SDA high to cease data transmission:



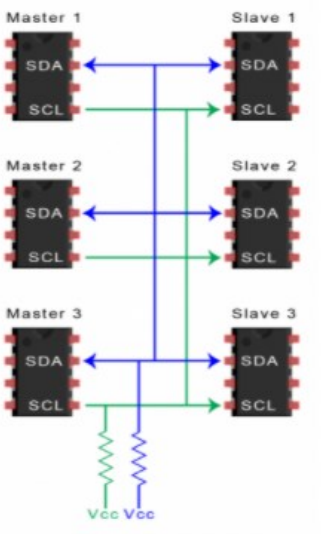
4.3.3 SINGLE MASTER MULTIPLE SLAVES

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



4.3.4 MULTIPLE MASTER MULTIPLE SLAVES

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



4.3.5 ADVANTAGES

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

4.3.6 DISADVANTAGES

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

4.4 SPI

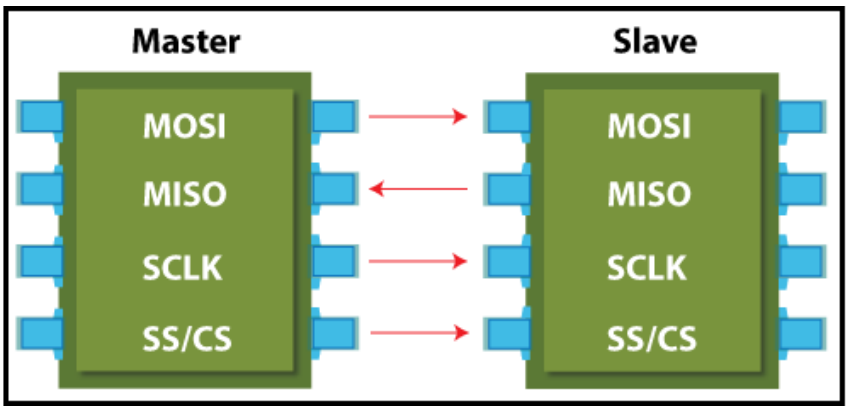
Serial Peripheral Interface (SPI) stands for Serial Peripheral Interface. It's a serial communication protocol used to link low-speed devices together. Motorola created it in the mid-1980s for inter-chip communication. It's frequently used to communicate with flash memory, sensors, real-time clocks (RTCs),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

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

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

4.4.1 SPI INTERFACE

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



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

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

4.4.2 CHARACTERISTICS OF SPI BUS

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

4.4.3 MULTI-DEVICE TOPOLOGIES

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

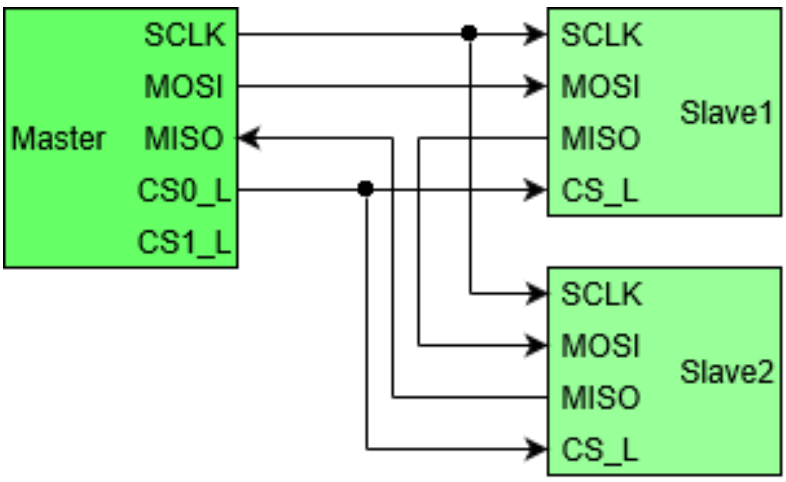


Figure 1.8 SPI Bus – Daisy Chain Topology

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

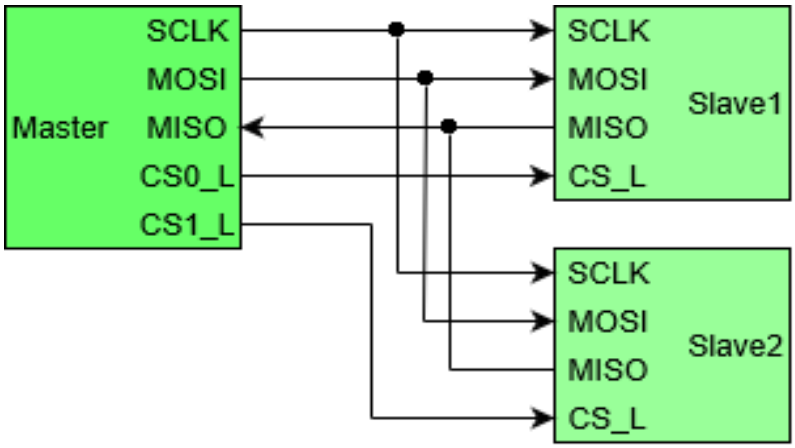


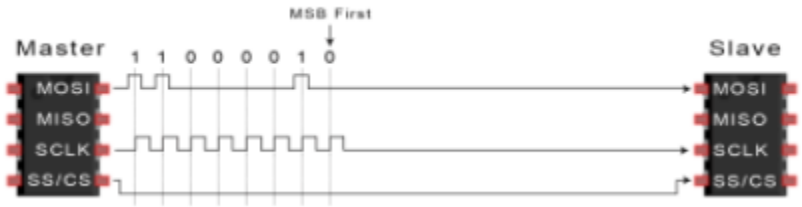
Figure 1.9 SPI Bus – Star Topology

4.4.4 SPI DATA TRANSMISSION STEPS

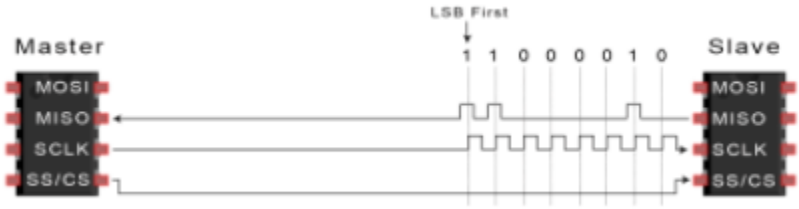
1. The clock signal is output by the master.

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

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



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



4.4.5 ADVANTAGES

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

4.4.6 DISADVANTAGES

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

4.4.7 APPLICATIONS

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

4.5 SUMMARY

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

4.6 LIST OF REFERENCES

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

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

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

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

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

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

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

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

4.7 UNIT END EXERCISES

1] Write a note on UART.

2] Explain the block diagram of UART and explain in brief why it is used?

3] State the advantages and disadvantages of UART.

4] Write a note on UART transmission steps.

5] Discuss the purpose, features of GPIO.

6] Write a note on working of I2C and state its advantages and disadvantages.

7] Describe the various I2C data transmission steps.

8] Write a note on single master multiple slaves.

9] Explain the concept of multiple master multiple slaves.

10] Write a note on SPI interfaces along with its characteristics.

11] Describe the SPI multidevice topologies.

12] Write a note on advantages, disadvantages and applications of SPI.

13] Explain the SPI data transmission steps.

❄❄❄❄❄❄❄

5

USEFUL IMPLEMENTATIONS

Unit Structure

5.0 OBJECTIVES

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

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

5.1 CROSS COMPILATION

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

The computer on which the compiler runs is known as the host, whereas the computer on which new programs execute is known as the target. The compiler is a native compiler when the host and target machines are of the same type. The compiler is a cross compiler when the host and target are different. The act of compiling code for one computer system (commonly referred to as the target) on a different computer system (often referred to

as the host) is known as cross-compilation. When the target system is too small to host the compiler and all essential files, this is a highly handy strategy.

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

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

5.1.1 NEED OF CROSS COMPILERS

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

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

5.1.2 WHY CROSS COMPILING IS DIFFICULT?

- **Portable native compiling is hard.**

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

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

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

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

- **Cross- compiling**

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

- **Configuration difficulties -** To be portable when natively compiled, packages having a separate configuration step (the "./configure" section of the typical configure/make/make install) frequently test for factors like endianness or page size. Because these values differ across the host and target systems when cross-compiling, performing tests on the host system yields incorrect results. When the target doesn't have that package or has an incompatible version, configuration can detect its presence on the host and include support for it.
- **HOSTCC vs. TARGETCC -** Many build procedures, such as the above configuration tests, or programs that generate code (such as a C program that generates a.h file that is then #included during the main build), need compiling items to execute on the host system. Simply substituting a target compiler for the host compiler damages packages that require the build of objects that run during the build process. These packages require access to both a host and a target compiler, as well as instruction on when to use each.

- **Toolchain Leaks -** An incorrectly configured cross-compile toolchain can leak pieces of the host system into built applications, causing failures that are normally easy to detect but complex to diagnose and fix. At link time, the toolchain may #include the incorrect header files or search the incorrect library directories. Shared libraries frequently rely on other shared libraries, which can introduce unintended host-system link-time references.

- **Libraries -** At compile time, dynamically linked applications must access the proper shared libraries. In order for programs to link against shared libraries on the target system, they must be included to the cross-compile toolchain.

- **Testing -** The development system provides a handy testing environment for native builds. Confirming that "hello world" compiled properly while cross-compiling can necessitate configuring (at the very least) a bootloader, kernel, root file system, and shared libraries.

5.1.3 WORKING OF CROSS COMPILATION

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

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

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

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

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

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

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

5.1.4 BUILD PROCESS OF CROSS COMPILER

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

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

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

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

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

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

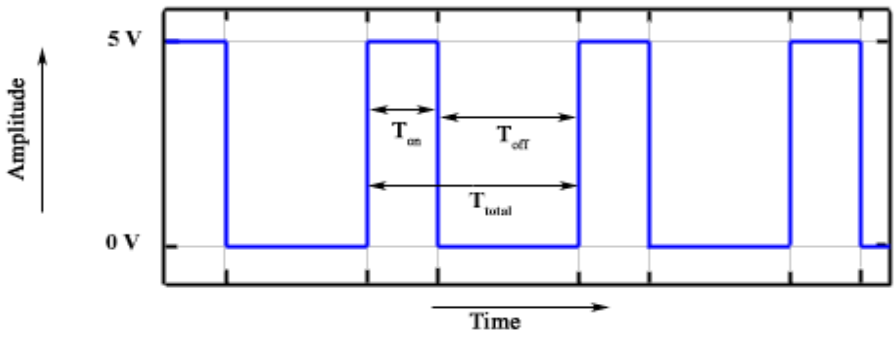
  You can also construct the GNU debugger - gdb - for the target host if you like. Because emulation is required, you can't create a debugger that can execute code for the target while running on the host. You can, however, create a gdb executable for your target host.

5.2 PULSE WIDTH MODULATION

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

5.2.1 PWM principle

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



Several terms are associated with PWM such as

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

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

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

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

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

- **Duty cycle:**

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

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

5.2.2 APPLICATIONS OF PWM

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

5.3 SPI FOR CAMERA

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

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

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

5.3.1 APPLICATIONS

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

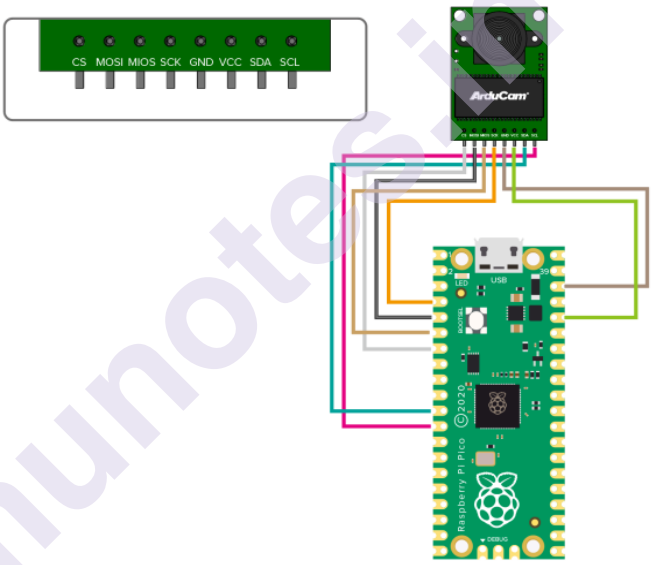
5.3.2 FEATURES

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

5.3.3 PIN DEFINITION

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

5.3.4 WIRING



Connect SPI Camera to Pico

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

5.4 SUMMARY

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

5.5 LIST OF REFERENCES

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

5.6 UNIT END EXERCISES

1] Explain the process of cross compilation.

2] What is the need of cross compilation?

3] Explain the working and build process of cross compilation.

4] State the principle of PWM.

5] State the features, applications and pin definition of SPI.

❄❄❄❄❄❄❄

6

IOT SERVICE AS A PLATFORM

Unit Structure

6.0 OBJECTIVE

- IoT requires expertise in enabling smart sensors to observe, learn, and make decisions to produce limitless market opportunities.
- IoT development services in India leverage widespread interconnected devices to enhance products that retain connectivity, convert data, and supervise them constantly.

6.1 INTRODUCTION

HTTP is a stateless request & response protocol where clients request information from a server and the server responds to these requests accordingly. A request is made up of a method, a resource, some headers, and some optional content. A response is made up of a three-digit status code, some headers and some optional content.

6.2 IOT SECURITY: HTTP, UPNP, COAP, MQTT, XMPP.

6.2.1. Hypertext Transfer Protocol (HTTP): It is used in machine-to-machine (M2M) communication, automation, and Internet of Things, among other things.

6.2.2. Introduction

HTTP is a stateless request & response protocol where clients request information from a server and the server responds to these requests accordingly. A request is made up of a method, a resource, some headers,

and some optional content. A response is made up of a three-digit status code, some headers, and some optional content. A Uniform Resource Locator (URL) identifies each resource, originally thought to be a collection of Hypertext documents or HTML documents. Clients simply use the GET method to request a resource from the corresponding server. In the structure of the URL presented next, the path and the server identify the resource by the authority portions of the URL. The PUT and DELETE methods allow clients to upload and remove content from the server, while the POST method allows them to send data to a resource on the server.

HTTP is a cornerstone of service-oriented architecture (SOA), where methods for publishing services through HTTP are called web services. One important manner of publishing web services is called Simple Object Access Protocol (SOAP), where web methods, their arguments, return values, bindings, and so on, are encoded in a specific XML format. It is then documented using the Web Services Description Language (WSDL).

6.2.3. Adding HTTP support to the sensor

The following are the three strategies used when publishing the data using HTTP:

- In the first strategy the sensor is a client who publishes information to a server on the Internet. The server acts as a broker and informs the interested parties about sensor values. This pattern is called publish/subscribe.
- Second Strategy is to allow all entities in the network be both clients and servers, depending on what they need to do. The UPnP Protocol. This reduces latency in communication but requires all participants to be on the same side of any firewalls.

Third Strategy is to let the sensor become an HTTP server, and who is interested in knowing the status of the sensor become the clients. It also allows easy access to the sensor from the parties behind firewalls if the sensor is publicly available on the Internet.

6.2.4. Setting up an HTTP server on the sensor

To begin with,

1. add the namespace in the application.

   using System.Xml;

   using System.Text;

   using System.IO;

   using System.Drawing;

2. Then, add references to the following Clayster namespaces, which will help to work with HTTP and along with different content types mentioned.

```
using Clayster.Library.Internet;
using Clayster.Library.Internet.HTTP;
using Clayster.Library.Internet.HTML;
using Clayster.Library.Internet.MIME;
using Clayster.Library.Internet.JSON;
using Clayster.Library.Internet.Semantic.Turtle;
using Clayster.Library.Internet.Semantic.Rdf;
using Clayster.Library.IoT;
using Clayster.Library.IoT.SensorData;
using Clayster.Library.Math;
```

The Internet library helps us with communication and encoding, the IoT library deals with an interoperability, and the Math library deals with graphs.

3. Next step is application initialization, which is done using the following code:

```
HttpSocketClient.RegisterHttpProxyUse (false, false);
```

To instantiate an HTTP server, add the following code before application initialization ends and the main loop begins:

```
HttpServer HttpServer = new HttpServer (80, 10, true, true, 1);
Log.Information ("HTTP Server receiving requests on port" + HttpServer.Port.ToString ());
```

The HTTP server can process both synchronous and asynchronous web resources:

A synchronous web resource responds within the HTTP handler we register for each resource. These are executed within the context of a working thread.

- An asynchronous web resource handles processing outside the context of the actual request and is responsible for responding by itself. This is not executed within the context of a working thread.

4. Register web resources on the server: register the path of each resource and connect that path with an HTTP handler method, which will process each corresponding request.

```
HttpServer.Register ("/", HttpGetRoot, false);
HttpServer.Register ("/html", HttpGetHtml, false);
HttpServer.Register ("/historygraph", HttpGetHistoryGraph, false);
HttpServer.Register ("/xml", HttpGetXml, false);
HttpServer.Register ("/json", HttpGetJson, false);
```

```
HttpServer.Register ("/turtle", HttpGetTurtle, false);
HttpServer.Register ("/rdf", HttpGetRdf, false);
```

5. Disposing the server when the application ends

```
HttpServer.Dispose ();
```

Adding a root menu

Add a root menu which is accessible through the path /.

```
private static void HttpGetRoot (HttpServerResponse resp, HttpServerRequest req)
{
 networkLed.High ();
 try
{
 resp.ContentType = "text/html";
 resp.Encoding = System.Text.Encoding.UTF8;
resp.ReturnCode = HttpStatusCode.Successful_OK;
}
finally
{
networkLed.Low ();
}
}
```

Return the actual HTML page with the following code:

```
resp.Write ("<html><head><title>Sensor</title></head>");
resp.Write ("<body><h1>Welcome to Sensor</h1>");
resp.Write ("<p>Below, choose what you want to do.</p><ul>");
resp.Write ("<li>View Data</li><ul>");
resp.Write ("<li><a href='/xml?Momentary=1'>");
resp.Write ("View data as XML using REST</a></li>");
resp.Write ("<li><a href='/json?Momentary=1'>");
resp.Write ("View data as JSON using REST</a></li>");
resp.Write ("<li><a href='/turtle?Momentary=1'>");
resp.Write ("View data as TURTLE using REST</a></li>");
resp.Write ("<li><a href='/rdf?Momentary=1'>");
resp.Write ("View data as RDF using REST</a></li>");
resp.Write ("<li><a href='/html'>");
```

```
resp.Write ("Data in a HTML page with graphs</a></li></ul>");
resp.Write ("</body></html>");
```

Accessing WSDL

The SOAP web service interface is documented in what is called a Web Service Definition Language (WSDL) document. The web services engine automatically generates this document.

Adding HTTP support to the controller

Sensor and an actuator that speaks about HTTP also need to add HTTP to the controller. The controller will act as an HTTP client.

6.3 THE UPNP PROTOCOL

Universal Plug and Play (UPnP) is a protocol or an architecture that uses multiple protocols, helps devices in ad hoc IP networks to discover each other, detects services hosted by each device, and executes actions and reports events. Ad hoc networks are networks with no predefined topology or configuration; here, devices find themselves and adapt themselves to the surrounding environment. UPnP is largely used by consumer electronics in home or office environments. The standard body for UPnP is the UPnP Forum (upnp.org). UPnP is largely based on an HTTP application where both clients and servers are participants.

Discovery of devices in the network is performed using Simple Service Discovery Protocol (SSDP), which is based on HTTP over UDP, and event subscriptions and notifications are based on General Event Notification Architecture (GENA). Both SSDP and GENA introduce new HTTP methods to search, notify and subscribe to and unsubscribe from an event. Actions on services are called using SOAP web service calls.

UPnP defines an object hierarchy for UPnP-compliant devices. Each device consists of a root device. Each root device can publish zero or more services and embedded devices. Each embedded device can iteratively publish more services and embedded devices by itself. Each service in turn publishes a set of actions and state variables. Actions are methods that can be called on the service using SOAP web service method calls. Actions take a set of arguments. Each argument has a name, direction (if it is input or output), and a state variable reference. From this reference, the data type of the argument is deduced. State variables define the current state of a service, and each one has a name, data type, and variable value. Furthermore, state variables can be normal, evented, and/or multicast-evented. When evented state variables change their value, they are propagated to the network through event messages. Normally, evented state variables are sent only to subscribers who use normal HTTP. Multicast-evented state variables are propagated through multicast HTTPMU NOTIFY messages on the SSDP multicast addresses being used, but using a different port number.

Each UPnP-compatible device in the network is described in a Device Description Document (DDD), an XML document hosted by the device

itself. When the device makes its presence known to the network, it always includes a reference to the location of this document. Interested parties then download the document and any referenced material to learn what type of device this is and how to interact with it. The document includes some basic information understandable by machines, but it also includes information for human interfaces. Finally, the DDD includes references to embedded devices, if any, and references to any services published by the device.

Each service published by a device is described in a standalone Service Control Protocol Description (SCPD) document, each one an XML document also hosted by the device. Even though SOAP is used to call methods on each service, UPnP-compliant services are drastically reduced in functionality compared to normal SOAP web services. SOAP and WSDL simply give devices too many options, making interoperability a problem.

6.4 THE COAP PROTOCOL

CoAP reduces the set of methods that can be used; it allows you to have four methods: GET, POST, PUT, and DELETE. In addition, in CoAP, method calls can be made using confirmable and non-confirmable message services. When you receive a confirmable message, the receiver always returns an acknowledgement. The sender can, in turn, resend messages if an acknowledgement is not returned within the given time period. The number of response code has also been reduced to make implementation simpler. CoAP also broke away from the Internet Media Type scheme used in HTTP and other protocols and replaced this with a reduced set of Content-Formats, where a number instead of its corresponding Internet Media Type identifies each format.

CoAP supports multicasting, which is used to detect devices or communicate through firewalls; it also provides a set of useful extensions. One of these extensions provides a block transfer algorithm, which allows you to transfer larger amounts of data. CoAP also supports encryption in the unicast case with Datagram Transport Layer Security (DTLS).

CoAP (resources)
UCP (ports)
Internet Protocol (unicast/multicast address)
Local Area Network (MAC address)
Physical (Cables, Radio)

Fig: CoAP protocol stack diagram:

CoAP is relatively new; the availability of development tools for this protocol is not available. There exists an add-on to Firefox, which allows you to view and interact with CoAP resources.

CoAP resources- The CoAP endpoint registers a resource by itself called .well-known/core. Here, it publishes a Link Format document called the Constrained RESTful Environments (CoRE) Link Format document. This document contains a list of resources published by the endpoint and some basic information about these documents. This document corresponds in some sense to WSDL documents for web services, even though the Link Format document is very lightweight. It consists of a sequence of resources and some corresponding attributes for each resource.

6.5 MQTT PROTOCOL

The MQTT protocol is based on the publish/subscribe pattern, as opposed to the request/response and the event subscription patterns. The publish/subscribe pattern has three types of actors:

a. Publisher: The role of the publisher is to connect to the message broker and publish content.

b. Subscriber: They connect to the same message broker and subscribe to content that they are interested in

c. Message broker: This makes sure that the published content is relayed to interested subscribers

Content is identified by topic. When publishing content, the publisher can choose whether the server should retain the content or not. If retained, each subscriber will receive the latest published value directly when subscribing. Furthermore, topics are ordered into a tree structure of topics, much like a file system. The forward slash character (/) is used as a delimiter when describing a topic path. When subscribing to content, a subscriber can subscribe to either a specific topic by providing its path, or an entire branch using the hash wildcard character (#). There is also a single-level wildcard character: the plus character (+).

<table>
<tr><td>MQTT</td><td>MQTT (SSL/TLS)</td></tr>
<tr><td>TCP
(port 1883)</td><td>TCP
(port 8883)</td></tr>
<tr><td colspan="2">Internet Protocol (IP)
(Unicast/multicast IP address)</td></tr>
<tr><td colspan="2">Local Area Network (LAN)</td></tr>
<tr><td colspan="2">Physical (Cables, Radio)</td></tr>
</table>

Fig : MQTT architecture

There are three Quality of Service levels in MQTT available while publishing content. The lowest level is an unacknowledged service. In this, the message is delivered at most once to each subscriber. The next level is an acknowledged service. Here, each recipient acknowledges the receipt of the published information. If no receipt is received, the information can be sent again. This makes sure the information is delivered at least once. The highest level is called the assured service. Here, information is not only acknowledged but sent in two steps. First, it is transmitted and then delivered. Each step is acknowledged. This makes it possible to make sure that the content is delivered exactly once to each subscriber.

6.6 XMPP PROTOCOL

Extensible Messaging and Presence Protocol (XMPP) protocol. The XMPP protocol also uses message brokers to bypass firewall barriers. Apart from the publish/subscribe pattern, it also supports other communication patterns, such as point-to-point request/response and asynchronous messaging, that allow you to have a richer communication

XMPP was originally designed for use in instant messaging applications (or chat). It is an open protocol, flexible and richness of communication patterns.

The XMPP architecture is built on scalability of the Simple Mail Transfer Protocol (SMTP). XMPP uses a network of XMPP servers as message brokers to allow clients behind separate firewalls to communicate with each other. Each server controls its own domain and authenticates users on that domain. Clients can communicate with clients on other domains using federation where the servers create connections between themselves in a secure manner to interchange messages between their domains. They only need to ensure that they maintain the connection with their respective servers, and through the servers, each of them will have the possibility to send messages to any other client in the federated network. XMPP is scalable and allows you to make billions of devices communicate with each other in the same federated network.

XMPP provides each client with an authenticated identity. When clients connect, the servers make sure the clients authenticate themselves by providing their corresponding client credentials, which would consist of a username and password. This authentication is done securely using an extensible architecture based on Simple Authentication and Security Layer (SASL). The connection can also be switched over to Transport Layer Security (TLS) through negotiation between the client and the server. The identity of the client is often called XMPP address or Jabber ID (JID).

The reason for using XMPP servers is to relay communication to assure the clients that only authorized communication will be relayed. This feature is used for small devices with limited decision-making capabilities. The server does so by ensuring that the full JID identifier instead of only the bare JID identifier is used to communicate with the application.

The reason is:

- First, multiple clients might use the same account at the same time. Then provide the resource part of the full JID for the XMPP Server to be able to determine which connection the corresponding message should be forwarded to. Only this connection will receive the message. This enables the actual clients to have direct communication between them.

- Second, only trusted parties (or friends) are given access to the resource part once the thing or application is connected. This means that, in turn, only friends can send messages between each other, as long as the resource parts are sufficiently long and random so they cannot be guessed, and the resource part is kept hidden and not published somewhere else. XMPP communication consists of bidirectional streams of XML fragments.

IoT Service as a Platform: Clayster, Thinger.io, SenseIoT, carriots and Node RED.

1. Clayster

There are many available platforms, they vary in functionality and development. To get the IoT platform go to http://postscapes.com/internet-of-things-platforms and review the registered platforms.

2. Clayster Platform

Download the Clayster platform by downloading from http://www.clayster.com/downloads. All the information about Clayster, including examples and tutorials, is available in a wiki. You can access this wiki at https://wiki.clayster.com/.

3. Libraries

Clayster.AppServer.Infrastructure: This library contains the application engine available in the platform. Apart from managing applications, it also provides report tools, cluster support, management support for operators and administrators; it manages backups, imports, exports, localization and various data sources used in IoT, and it also provides rendering support for different types of GUIs, among other things.

Clayster.Library.Abstract: This library contains a data abstraction layer, and is a crucial tool for the efficient management of objects in the system.

Clayster.Library.Installation: This library defines the concept of packages.

Clayster.Library.Meters: This library replaces the Clayster. Library. IoT library used in previous chapters. It contains an abstraction model for things such as sensors, actuators, controllers, meters, and so on.

Clayster: To facilitate the development of IoT applications, seven Clayster libraries are used for private and commercial applications.

Clayster Library	**Description**
Clayster.Library.Data	It provides the application with a powerful object database. Objects are persisted and can be searched directly in the code using the object's class definition. Data can be stored in the SQLite database provided in Raspberry Pi.
Clayster.Library.EventLog	This provides the application with an extensible event logging architecture that can be used to get an overview of what happens in a network of things.
Clayster.Library.Internet	It consist of classes that implements various internet protocol. This library can be used dynamically for communication via internet.
Clayster.Library.Language	It is used to create localizable applications that are simple to translate and that can work in an international setting.
Clayster.Library.Math	It has powerful mathematical scripting language that will be used in automation, scripting, graph plotting, and others.
Clayster.Library.IoT	It has classes that help the applications to become interoperable by providing data representation and parsing capabilities of data in IoT.
Clayster.Library.RaspberryPi	It has Hardware Abstraction Layer (HAL) used in Raspberry Pi. It provides object-oriented interfaces to interact with devices connected to the general-purpose Input/output (GPIO) pins available.

4. Service modules

The service modules available are:

Clayster.HomeApp.MomentaryValues: This is a simple service that displays momentary values using gauges. We will use this project to display gauges of our sensor values.

Clayster.Metering.Xmpp: This module contains an implementation of XMPP on top of the abstraction model defined in the Clayster.Library.Metersz namespace.

5. Clayster Management Tool (CMT)

It comes with a management tool that helps you to manage the server. This Clayster Management Tool (CMT) can also be downloaded from http://www.clayster.com/downloads. This includes data sources, objects in the object database, current activities, statistics and reports, and data in readable event logs

Browsing data sources

Most of the configurable data in Clayster is ordered into data sources. These can be either tree-structured, flat or singular data sources. Singular data sources contain only one object. Flat data sources contain a list (ordered or unordered) of objects. Tree structured data sources contain a tree structure of objects, where each object in the structure represents a node. The tree-structured data sources are the most common, and they are also often stored as XML files. Objects in such data sources can be edited directly in the corresponding XML file, or indirectly through the CMT, other applications or any of the other available APIs. When you open the CMT for the first time, make sure that you open the Topology data source. It is a tree-structured data source whose nodes represent IoT devices. The tree structure shows how they are connected to the system. The Root represents the server itself.

Interfacing the devices

XMPP is already implemented and supported through the Clayster. Metering. This module models each entity in XMPP as a separate node in the Topology data source. Connections with provisioning servers and thing registries are handled automatically through separate nodes dedicated to this task. Friendships are handled through simple child creation and removal operations. It can be done automatically through requests made by others or recommendations from the provisioning server, or manually by adding friends in the CMT. Provide specialized classes that override base class functionality and add specific features that are needed.

6. Thinger.io

What is Thinger.io

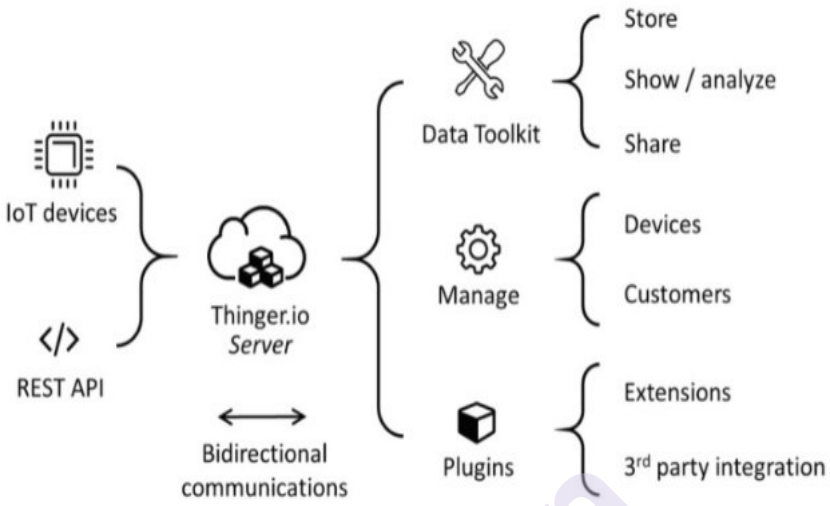
Thinger.io is a cloud IoT Platform that provides every needed tool to prototype, scale and manage connected products in a very simple way.

Features

- **Free IoT platform**: Thinger.io provides a lifetime freemium account with only few limitations to start learning and prototyping when your product becomes ready to scale, you can deploy a Premium Server with full capacities within minutes.
- **Simple but Powerful**: Just a couple code lines to connect a device and start retrieving data or controlling its functionalities with our web-based Console, able to connect and manage thousands of devices in a simple way.
- **Hardware agnostic:** Any device from any manufacturer can be easily integrated with Thinger.io's infrastructure.
- **Extremely scalable & efficient infrastructure:** thanks to our unique communication paradigm, in which the IoT server subscribes device resources to retrieve data only when it is necessary, a single Thinger.io instance is able to manage thousands of IoT devices with low computational load, bandwidth and latencies.
- **Open-Source**: most of the platform modules, libraries and APP source code are available in our Github repository to be downloaded and modified with MIT license.

  Thinger.io platform is formed by two main products a Backend (which is the actual IoT server) and a web-based Frontend that simplifies working with all the features using any computer or smartphone.
- **Connect devices:** Fully compatible with every kind of device, no matter the processor, the network, or the manufacturer. Thinger.io allows to create **bidirectional communications** with Linux, Arduino, Raspberry Pi, or MQTT devices and even with edge technologies like Sigfox or LoRaWAN or other internet API data resources.
- **Store Device Data:** Just a couple clicks to create a Data Bucket a store IoT data in a scalable, efficient, and affordable way, that also allows real-time data aggregation.
- **Display Real-time** or **Stored Data** in multiple widgets such as time series, donut charts, gauges, or even custom-made representations to create awesome dashboards within minutes.
- **Trigger events and data values** using an embedded Node-RED rule engine
- **Extend with custom features** with multiple plugins to integrate IoT projects into your company's software or any other third-party Internet service.

- **Custom the appearance** thanks to our fully rebrand able frontend, that allows introducing your branding colours, logotypes and web domain. Refer: https://docs.thinger.io/



7. SenseIoT

The Sense IoT cloud server is a **logical server** built, hosted, and delivered through a cloud computing platform over the internet. Unlike normal physical servers, cloud servers can be accessed remotely at any time.

The term IoT stands for Internet of Things, and it is the most significant as well as promising technology nowadays. There are a billion devices relate to sensors like wearables, smartphones, etc. Currently, every sensor plays an essential role in the Internet of Things. These sensors are mainly used for detecting or monitoring the quality of air, health status, home security, etc. Similarly, these sensors are used in IoT for monitoring the process of production, so named as IoT sensor.

There are different types of sensors which is used for different applications like to collect the data from the environment. In an IoT ecosystem, there are two main things the internet & the physical devices such as actuators & sensors. The sensor and network connectivity in the IoT mainly located in the bottom layer. The main function of this is to collect the information. This bottom layer in the IoT is a very important part, and it includes connectivity of network to next layer like the gateway & network layer.

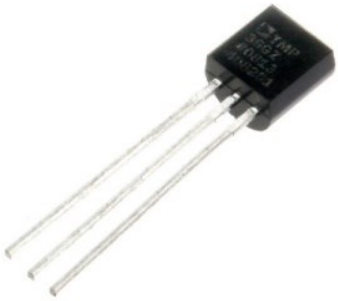
The main function of these sensors is to gather information from the surroundings. The connection of these to IoT can be done directly otherwise indirectly once the conversion of signal & processing is done. All the sensors are not similar because different IoT applications need different kinds of sensors.

Types of IoT Sensors

The different types of IoT sensors with its working as follows:

Temperature Sensor

The temperature sensor is used to detect the heat energy which is produced from an object or nearby area. The main role of these sensors in manufacturing is for temperature monitoring of machines. Similarly, in the agriculture field, these sensors are used to monitor the temperature of plants, soil, and water. The applications of temperature sensors mainly include refrigerators, ACs, etc.



Smoke Sensor

Smoke sensors have been using in various applications like homes, industries, etc. These sensors are very convenient as well as easy to use by the arrival of the Internet of Things. Also, by adding a wireless connection to smoke detectors, the additional features can be enabled to increase security & ease.

Motion Sensor

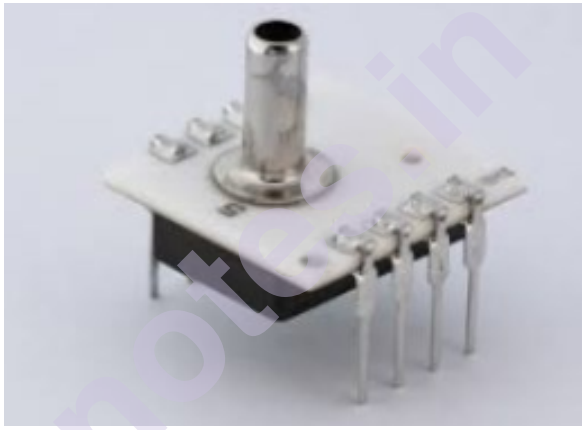
The motion sensor is used in hand dryers, energy management systems, automatic parking systems, automatic door controls, automated toilet flushers, automated sinks, etc.

Humidity Sensors

Humidity sensors are used to monitor the level of humidity in the amount of vapor of water within the air. The units for measurement humidity is RH (relative humidity) & PPM (parts per million).

Pressure Sensor

The pressure sensors are used in IoT for monitoring devices and systems which are determined by force signals. As the range of pressure is outside the threshold stage, then the device gives an alert to the user regarding the issues that must be fixed. The best example of a pressure sensor is BMP180, which can be used in mobile phones, GPS navigation devices, etc. These sensors are also applicable in aircraft and smart vehicles to decide altitude & force correspondingly. In a motor vehicle, TMPS (tire pressure monitoring system) can also be used for giving an alert to the driver while tire pressure is extremely less & it could make unsafe driving situations.



Gas Sensor

Gas sensors are mainly used to detect toxic gases. The most frequently used technologies are photoionization, semiconductor, and electrochemical.

IR Sensors

Infrared sensors are mainly used to measure the heat which is produced by objects. These sensors are used in the various applications of IoT like healthcare for monitoring the flow of blood, BP, etc. These sensors are used in smartphones for controlling, wearable devices for detecting the amount of light, detection of blind spot within vehicles, etc.

Accelerometer Sensor

Accelerometer sensors are utilized in aircrafts vehicles, smartphones. Similarly, these are used in different applications to identify the direction of an object, tilt, tap, shake, positioning, and motion, vibration, or shock. Types of accelerometers are like capacitive, Hall-effect & piezoelectric.

Image Sensor

Image sensors are applicable in medical imaging systems, thermal imaging devices, digital cameras, night-vision equipment, sonars, radars, & biometric systems. These sensors are used in the retail industry for monitoring the visiting count of the customers in the store with the help of network like IoT. The applications of image sensors mainly include offices, corporate buildings for monitoring the employees.

Proximity Sensors

Proximity sensors are used to detect the existence or nonexistence of a near object with no physical contact. These sensors are classified into different types like capacitive, inductive, ultrasonic, magnetic, and photoelectric. These sensors are frequently used for process monitoring, control, and object counters.

Source: IoT Sensor: Different Types, Working and Its Applications (elprocus.com)

7. Carriots

The Carriots IoT Platform is designed to help companies, businesses and institutions to improve their business and efficiency by controlling their IoT devices, and provides safe transfer of information among IoT devices, cloud and by bringing solutions to the problems by using AI and other technologies.

Carriots is a smart Platform as a Service (PaaS), functions in a machine-to-machine way and doesn't necessitate any human interruption after the system is set up. It is also established by Altair Engineering.

What Does Carriots Do?

Carriots has a four-staged working mechanism in helping a business to find solutions to its problems. First stage is the data analysis stage. The investors who want to develop their business by innovation gives the data to the Carriots IoT platform after it is set up to the system. After the data on business is given to the Carriots platform, the Carriots begins to analyse the data and try to detect the problems, obstacles and other latent issues in the business operation process. In the second stage, Carriots tries to find a solution to the problems that it detected in the first stage. It has its own mechanism to eliminate the methods which are not useful to the business. In the third stage, it approves the solution method which should be implemented on the business that decided in the second stage.

What are the Advantages of Carriots IoT Platform?

Features of Carriots that make it more advantageous:

- Carriots has an open architecture that enables it to work in collaboration with the third-party machines. This enables it to find the

best solution due to its wide data collection because of this feature and flexibility.

- The Carriots Platform can be controlled by a device that is connected to a remote controller. This feature enables to keep a watch what they are doing with your business. Also, Carriots is not bound by the check status and can change configurations.
- Rules: Carriots has rule to apply in complex business scripts.
- Carriots also has new triggers that push the data into the platforms and enables the usage of that data.
- It can integrate with other AI and IT systems
- System also has custom alarms when it faces a problematic situation during the innovation.

Source: Carriots IoT Platform | IoT5.net

9.Node RED.

Node-Red is based on graphical interface and has three components:

Flow -A project is called a flow and consist of data and functions linked together.

Message-It carries data from one node to another.

Nodes- These are the functions that generate, transfer or use messages.

The Node-RED GUI consists of three parts, from left to right:

- The left pane lists all the nodes, grouped by categories.
- The centre pane corresponds to the working area, where the flow is going to be designed.

- The right pane provides useful tools as documentation, a console for debugging, and the organisation for the dashboard.

Node-RED offers native support for other services. For example, there is a node for sending e-mails. Node-RED relies on MQTT, which requires a TCP/IP stack.

6.7 SUMMARY

- Innovative companies utilize information potential in Internet of Things as a service (IoTaaS) technology that senses information to create better products for that satisfies customers.
- The complex handling of IoT's fusion technologies leaves enterprises without in-house expertise or software to make IoT a profitable and worthwhile investment.

6.8 REFERENCES

- Learning Internet of Things by Peter Waher, Packt Publishing. (All notes are taken from this prescribed reference book).
- Notes also taken from the link below:
- https://github.com/Clayster/ Learning-IoT-CoAP.
- https://docs.thinger.io/
- IoT Sensor: Different Types, Working and Its Applications (elprocus.com)

6.9 UNIT END EXERCISE QUESTIONS

1. What is Clayster in IoT? Explain different libraries in it.
2. Explain the following IoT security: HTTP, UPnp, CoAP, MQTT, XMPP
3. What is meant by IoT service as a platform?
4. Explain different types of IoT Service as a Platform: Clayster, Thinger.io, SenseIoT, carriots and Node RED.
5. What is Thinger.io? What are the features of it.
6. Describe SenseIoT?
7. What are sensors? Explain different types of sensors used in IoT.
8. What are Carriots? What are its advantages

❄❄❄❄❄❄❄

7

IOT SECURITY AND INTEROPERABILITY

Unit Structure

7.1 OBJECTIVE

Hardware, Software, and other devices needs security to work efficiently. Without security in IoT devices & it's hardware will result in hacking & hackers will gain access to the controlling of the IoT devices and hence the IOT application will fail.

This chapter talks about the different types of security mechanisms, protocols, methods & techniques that should be deployed when the IoT devices & its application are ready to function.

Also discussed various types of virus, malwares & Trojan horse which has harmed the devices & made the resources less to perform.

7.2 INTRODUCTION

There are a lot of different technologies that can be used for Internet of Things (IoT), but security and interoperability issues come to any extent. We will discuss the topics, issues and that need to be addressed during the design of the overall architecture to avoid many of the unnecessary problems that might otherwise arise and minimize the risk.

7.3 IOT SECURITY AND INTEROPERABILITY

Risks with IoT, Modes of attacking a system and some counter measures, The importance of interoperability in IoT.

7.4 RISKS

There are many solutions and products under IoT that lack basic security architectures. It is very easy for a knowledgeable person to take control of devices for malicious purposes. Not only devices at home are at risk, but cars, trains, airports, stores, ships, logistics applications, building automation, utility metering applications, industrial automation applications, health services, and so on, are also at risk because of the lack of security measures in their underlying architecture.

5.MODES OF ATTACKS

a. Denial of Service

A Denial of Service (DoS) or Distributed Denial of Service (DDoS) attack is normally used to make a service on the Internet crash or become unresponsive, and in some cases, behave in a way that it can be exploited. The attack consists in making repetitive requests to a server until its resources gets exhausted. In a distributed version, the requests are made by many clients at the same time, which obviously increases the load on the target. It is often used for blackmailing or political purposes.

b. Guess the Username and Password

Getting it in the system is to try to impersonate by guessing the username and password of the authenticated clients. Guessing the client credentials is risky and it makes the attack less effective, in the communication. Always have the habit of changing the credentials frequently. The preset and fixed password helps the attacker to crack it easily.

c. Access to stored credentials

People reusing the credentials in different systems. There are various ways to avoid this risk. Credentials of the clients are not to be reused in different devices or across different services and applications. Another is to randomize the credentials, lessening the desire to reuse memorized credentials. Never store actual credentials centrally, even encrypted if possible, and instead store hashed values of these credentials. This is often possible since authentication methods use hash values of credentials in their computations. Even though some hashing functions are vulnerable in such a way that a new string can be found that generates the same hash value.

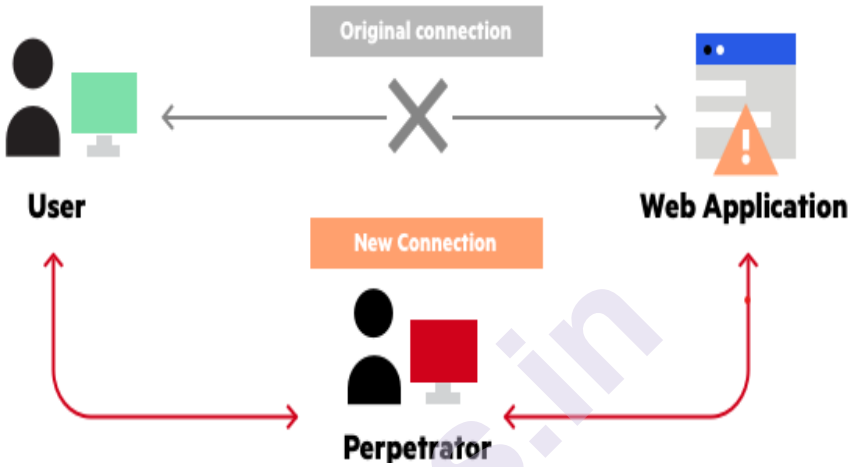
d. Man in the Middle attack (MITM)

Man-in-the-middle (MITM) attacks are a valid and extremely successful threat vector.

A man in the middle (MITM) attack is a general term for when a perpetrator positions himself in a conversation between a user and an

application—either to eavesdrop or to impersonate one of the parties, making it appear as if a normal exchange of information is underway.

The goal of an attack is to steal personal information, such as login credentials, account details and credit card numbers. Targets are typically the users of financial applications, businesses, e-commerce sites and other websites where logging is needed.



https://www.imperva.com/learn/application-security/man-in-the-middle-attack-mitm/

An MITM attack can take a few different forms. ARP poisoning is the most common, but DHCP, DNS, and ICMP poisoning are also effective, as well as the use of a malicious wireless access point (AP). Fake APs have become a common threat vector, exploiting the way clients automatically connect to known SSIDs. This enables an attacker to connect and intercept the victim's network traffic without the victim seeing any indication they are under attack. To hasten a connection, attacks against the legitimate AP can be made to help the malicious AP become the last AP standing.

It can lead to launch the different attacks:

IP spoofing involves an attacker disguising himself as an application by altering packet headers in an IP address. As a result, users attempting to access a URL connected to the application are sent to the attacker's website.

ARP spoofing is the process of linking an attacker's MAC address with the IP address of a legitimate user on a local area network using fake ARP messages. As a result, data sent by the user to the host IP address is instead transmitted to the attacker.

DNS spoofing, also known as DNS cache poisoning, involves infiltrating a DNS server and altering a website's address record. As a result, users attempting to access the site are sent by the altered DNS record to the attacker's site.

MITM Attack Prevention

- Avoiding Wi-Fi connections that aren't password protected.
- Paying attention to browser notifications reporting a website as being unsecured.
- Immediately logging out of a secure application when it's not in use.
- Not using public networks (e.g., coffee shops, hotels) when conducting sensitive transactions.

e. Sniffing Network Communications

Sniffing occurs when an unauthorized third party captures network packets destined for computers other than their own. Packet sniffing allows the attacker to look at transmitted content and may reveal passwords and confidential data.

Specialized packet driver software must be connected to the network segment they want to sniff, and must use sniffer software. By default, a network interface card (NIC) in a computer will usually drop any traffic not destined for it. By putting the NIC in promiscuous mode, it will read any packet going by it on the network wire.

Packet-sniffing attacks are more common in areas where many computer hosts share the same collision domain.

Sniffing and Spoofing-Computer's exchange messages with each other in the form of small groups of data called packets. Packets contains the actual data to be sent & addressing information. Attackers target these packets as they travel from source to destination. Thus, two types of attacks:

Packet Sniffing(snooping)/IP Sniffing – It is a type of passive attack where an attacker need not hijack a conversation but simply observe(sniff) packets as they pass through. To prevent sniffing, the information must be protected by encoding or the transmission link itself is encoded.

Packet Spoofing/IP Spoofing – The attacker sends packets with an incorrect source address. The receiver will receive these packets containing false address and replies back to this forged address (spoofed address) not to an attacker. It will lead to three possible cases:

- If the attacker is between the destination and the forged source, the attacker can see the reply & use that information for hijacking.
- If the attacker's intention was DOS attack, then the attacker need not bother about the reply.

- The attacker could simply be angry with the host, so it may put that host's address as the forged source address & send the packet to the destination. The attacker does not want a reply from the destination, as it wants the host with the forged address to receive it & get confused.

Pharming (DNS Spoofing)-

Domain Name Server maintains the mapping between domain names & the corresponding IP address. Eg- Merchant (Bob) whose sites domain name is **Error! Hyperlink reference not valid.** & The IP address is 100.10.10.20 in the DNS.

Attacker (Tom IP address- 100.20.20.20) manages to hack and replace the IP address of the Bob with its own IP address. Now Bob's IP address is 100.20.20.20 for **Error! Hyperlink reference not valid.**.

Alice will communicate with Bob's site; it will query the DNS & will get Bob's IP address - 100.20.20.20. Hence Alice is communicating with an attacker (Tom) assuming that she is communicating with the Bob.

f. Port Scanning and Web Crawling

It is technique of determining which port on the network is open and which could send and receive data, it is also a process for sending packets to specific ports on a host and analyzing responses to identify vulnerabilities. This scanning take place by first identifying a list of active hosts and mapping those hosts to their IP addresses. The goal behind port and network scanning is to identify the organization of IP addresses, hosts, and ports to properly determine open or vulnerable server locations and diagnose security levels. Port scanning can be used by both IT administrators and cybercriminals to verify or check the security policies of a network and identify vulnerabilities, also exploit any potential weak entry points.

Web Crawling Web crawling is the process of indexing data on web pages by using a program or automated script. These automated scripts or programs are known by multiple names, including web crawler, spider, spider bot, or a crawler.

Web crawlers copy pages for processing by a search engine, which indexes the downloaded pages so that users can search more efficiently. The goal of a crawler is to learn what webpages are about. This enables users to retrieve any information on one or more pages when it's needed.

g. Wildcards

Devices that use multicast communications, such as UPnP, CoAP, anybody can listen and see who sends the messages. Devices that use single-cast communication, such as those using HTTP or CoAP, port-

scanning techniques can be used. For devices that are protected by firewalls and use message brokers to protect against incoming attacks, such as those that use XMPP and MQTT, search features or wildcards can be used to find the identities of devices managed by the broker

HTTP and CoAP that support some level of local client authentication but lacks a good distributed identity and authentication mechanism.

XMPP only permits messages from specific requests that has come from another end. The device which accepts the request is a friend devise or not. This can be either configured by somebody else with access to the account or by using a provisioning server if the device cannot make such decisions by itself. The device does not need to worry about client authentication, as this is done by the brokers themselves, and the XMPP brokers always propagate the authenticated identities of everybody who send them messages.

In MQTT the only way to control who gets access to the data is by building a proprietary end-to-end encryption layer on top of the MQTT protocol, thereby limiting interoperability.

h. Breaking Ciphers

Cryptanalysis is the study of analyzing information systems in order to study the hidden aspects of the systems. Cryptanalysis is used to breach cryptographic security systems and gain access to the contents of encrypted messages, even if the cryptographic key is unknown.

In addition to mathematical analysis of cryptographic algorithms, cryptanalysis includes the study of side-channel attacks that do not target weaknesses in the cryptographic algorithms themselves, but instead exploit weaknesses in their implementation. Attacks can be classified based on what type of information the attacker has available.

Cryptography-An art of achieving security by encoding messages to make them non-readable.

Cryptanalysis- A technique of decoding messages from a non-readable format back to a readable format without knowing how they were initially converted from readable format to a non-readable format.

Plain Text – Message in a plain text can be understood by anybody & it is in human readable form.

Cipher Text –When a plain text message is coded using a suitable scheme, the resulting message is called cipher text.

Encryption(encoding)- It transforms a plain text into a cipher text.

Decryption(decoding) – It transforms a cipher text into a plain text.

Two ways in which a plain text can be coded to obtain the corresponding cipher text - substitution & Transposition.

Eg :Substitution Techniques – The characters of a plain text messages are replaced by other characters, numbers or symbols.

a. **Caesar Cipher – Algorithm**

- Read each alphabet in the cipher text message, & search for it in the second row of the replacement table.
- When a match is found replace that alphabet in the cipher text message with the corresponding alphabet in the same column but the first row of the table
- Repeat the process for all alphabets in the cipher text message.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Eg- Plain Text – HELLO HOW ARE YOU

Cipher Text – KHOOR KRZ DUH BRX.

Eg- Cipher Text – I AM FINE

Plain Text - F XJ CFKB

I. Password Cracking

Password crackers either try to guess passwords or they use brute-force tools. *Brute-force* tools attempt to guess a password by trying all the character combinations listed in an accompanying *dictionary*. The dictionary may start off blindly guessing passwords using a simple incremental algorithm. (For example, trying aaaaa, aaaab, aaaac, and so on) or it may use passwords known to be common on the host (such as password, blank, michael, and so on).

If the attacked system locks out accounts after a certain number of invalid login attempts, some password attackers will gain enough access to copy down the password database, and then brute force it offline.

7.6 TOOLS FOR ACHIEVING SECURITY

There are a number of tools that architects and developers can use to protect against malicious use of the system.

a. Virtual Private Networks

A method that is often used to protect unsecured solutions on the Internet is to protect them using Virtual Private Networks (VPNs). Machine to Machine solutions work well in local intranets that needs to expand across the Internet. One way to achieve this is to create such VPNs that allow the devices to believe they are in a local intranet, even though communication is transported across the Internet. Telephone operators use the

Internet to transport long distance calls, it doesn't make it Voice over IP (VoIP). Using VPNs might protect the solution, but it eliminates the possibility to interoperate with others on the Internet.

A virtual private network, or VPN, is an encrypted connection over the Internet from a device to a network. The encrypted connection helps ensure that sensitive data is safely transmitted. It prevents unauthorized people from eavesdropping on the traffic and allows the user to conduct work remotely. VPN technology is widely used in corporate environments.

How does a virtual private network (VPN) work?

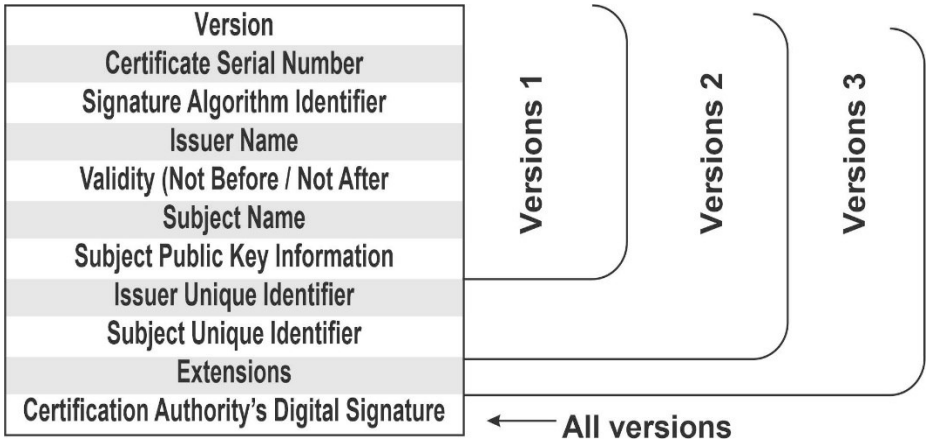
A VPN extends a corporate network through encrypted connections made over the Internet. Because the traffic is encrypted between the device and the network, traffic remains private as it travels. An employee can work outside the office and still securely connect to the corporate network. Even smartphones and tablets can connect through a VPN.

b. X.509 certificates and encryption

The use of certificates to validate the identity of high-value entities on the Internet. Certificates allow you to validate not only the identity, but also to check whether the certificate has been revoked or any of the issuers of the certificate have had their certificates revoked, which might be the case if a certificate has been compromised. Certificates also provide a Public Key Infrastructure (PKI) architecture that handles encryption. Each certificate has a public and private part. The public part of the certificate can be freely distributed and is used to encrypt data, whereas only the holder of the private part of the certificate can decrypt the data. Using certificates incurs a cost in the production or installation of a device or item. They also have a limited life span, so they need to be given either a long lifespan or updated remotely during the life span of the device. Certificates also require a scalable infrastructure for validating them. It is difficult to see that certificates will be used by other than high-value entities that are easy to administer in a network.

Digital Certificate

A standard called X.509 defines the structure of a digital certificate. Figure shows the structure of a X.509 V3 digital certificate.



Field	Description
Version	Identifies a particular version of the X.509 protocol, which is used for this digital certificate. Currently, this field can contain 1, 2 or 3.
Certificate Serial Number	Contains a unique integer number, which is generated by the CA.
Signature Algorithm Identifier	Identifies the algorithm used by the CA to sign this certificate.
Issuer Name	Identifies the Distinguished Name (DN) of the CA that created and signed this certificate.
Validity (Not Before/Not After)	Contains two date-time values (Not Before and Not After), which specify the time frame within which the certificate should be considered valid. These values generally specify the date and time up to seconds or milliseconds.
Subject Name	Identifies the Distinguished Name (DN) of the end entity (i.e. the user or the organization) to whom this certificate refers. This field must contain an entry unless an alternative name is defined in Version 3 extensions.
Subject Public Key Information	Contains the subject's public key and algorithms related to that key. This field can never be blank.

PKIX Services

a. Registration

It is the process where an end-entity (subject) makes itself known to a CA. Usually, this is via an RA.

b. Initialization

This deals with the basic problems, such as who the end-entity is sure that it is talking to the right CA?

c. Certification

In this step, the CA creates a digital certificate for the end-entity and returns it to the end-entity maintains a copy for its own records and copies it in public directories, if required.

d. Key pair recovery

Keys used for encryption may be required to be recovered later for decrypting some old documents. Key archival and recovery services can be provided by a CA or by an independent key recovery system.

e. Key generation

PKIX specifies that the end-entity should be able to generate private and public key pairs or the CA/RA should be able to do this for the end-entity (and then distribute these keys securely to the end-entity).

f. Key update

This allows a smooth transition from one expiring key pair to a fresh one, by the automatic renewal of digital certificates. However, there is a provision for manual digital certificates renewal request and response.

g. Cross-certification

Helps in establishing trust models, so that end-entities that are certified by different CAs can cross-verify each other.

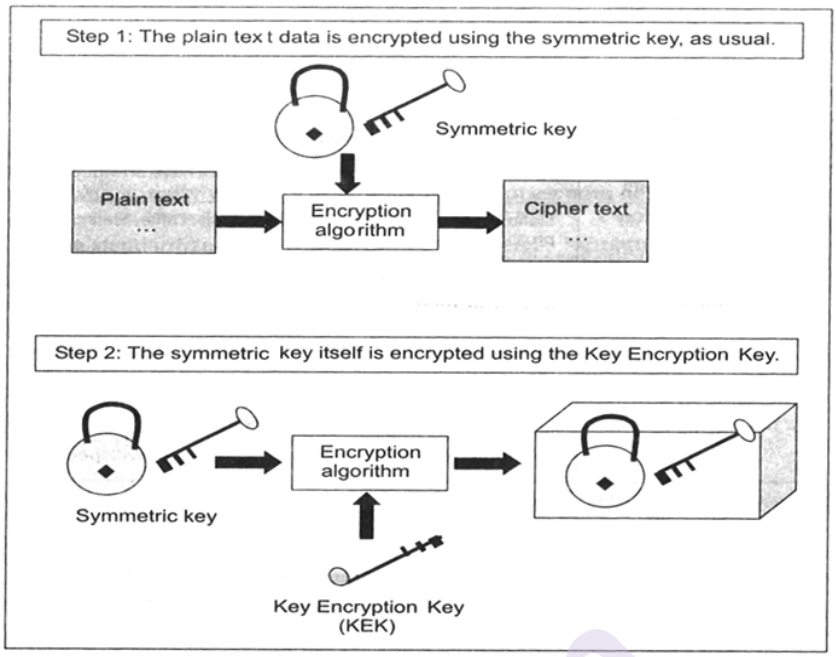
h. Revocation

PKIX provides support for the checking of the certificate status in two modes: online (using OCSP) or offline (using CRL).

Public Key Cryptography Standards (PKCS)

PKCS#5 ¾ Password Based Encryption (PBE) Standard

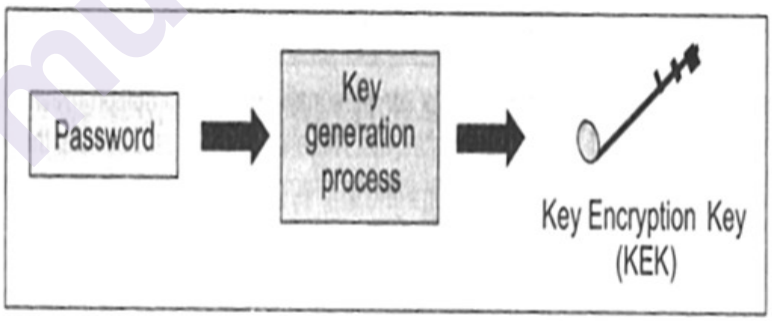
PBE is a solution for keeping the symmetric session keys safe. This technique ensures that the symmetric keys are protected from an unauthorized access. The PBE method uses a password-based technique for encrypting a session key.



first encrypt the plain-text message with the symmetric key, and then encrypt the symmetric key with a Key Encryption Key (KEK). This protects the symmetric key from an unauthorized access.

To protect KEK, never store it anywhere! This will ensure that no one will have access to it.

the approach used in PBE is to generate it on demand, use it for encrypting/decrypting the symmetric key, and then discard it immediately. The password is the input to a key-generation process (usually a message digest algorithm), the output of which is the KEK.



c. Authentication of identities

Authentication is the process of validating whether the identity provided is actually correct or not. Authenticating a server might be as simple as validating a domain certificate provided by the server, making sure it has not been revoked and that it corresponds to the domain name used to connect to the server. Authenticating a client might be more involved, as it has to authenticate the credentials provided by the client. Normally, this can be done in many different ways. It is vital for developers and architects to understand the available authentication methods and how they work to be able to

assess the level of security used by the systems they develop. Some protocols, such as HTTP and XMPP, use the standardized Simple Authentication and Security Layer (SASL) to publish an extensible set of authentication methods that the client can choose from. This is good since it allows for new authentication methods to be added. But it also provides a weakness: clients can be tricked into choosing an unsecure authentication mechanism, thus unwittingly revealing their user credentials to an impostor. MD5-DIGEST, and so on, even if they are the only options available.

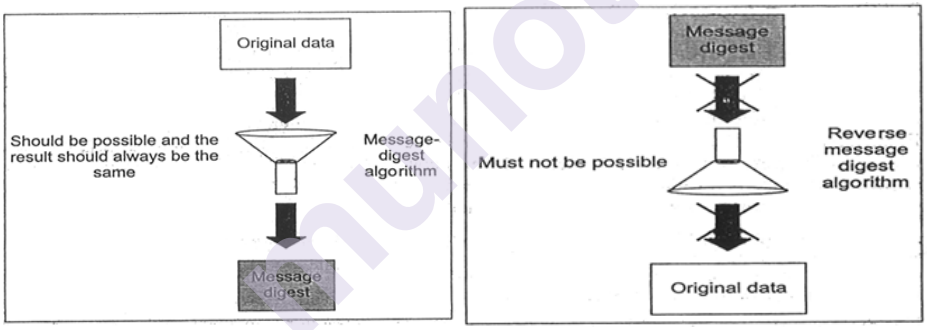
Message Digest

What is a message digest -It is a fingerprint or the summary of a message? It is similar to the concept of Longitudinal Redundancy Check (LRC) or Cyclic Redundancy Check (CRC) that is used to verify the integrity if the data. (i.e., to ensure that a message has not been tampered with after it leaves the sender before it reaches the receiver)

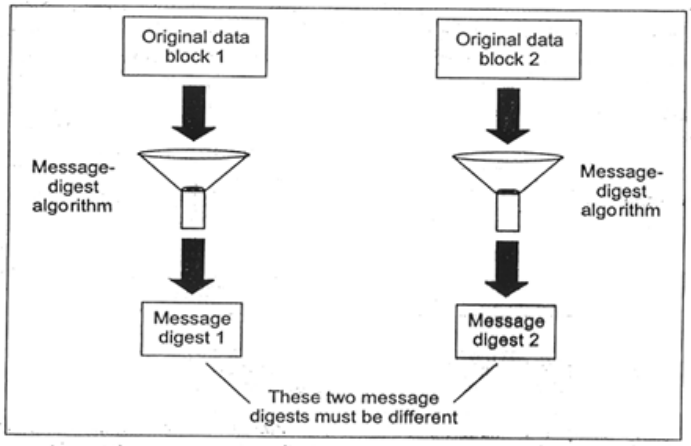
The requirements of the message digest concept, as follows:

(a) Given a message, it should be very easy to find its corresponding message digest. This is shown in Figure, Also, for a given message, the message digest must always be the same.

(b) Given a message digest, it should be very difficult to find the original message for which the digest was created.



(c) Given any two messages, if we calculate their message digests, the two message digests must be different.



MD5 hashing algorithm:

MD5 is quite fast and produces 128-bit message digest.

Secure Hash Algorithm (SHA).

SHA works with any input message that is less than 2 bits in length. The output of SHA is a message digest, which is 160 bits in length (32 bits more than the message digest produced by MD5).

Comparison between MD5 & SHA.

Points	MD5	SHA
Message-digest length in bits	128	160
Attack to try and find the original message given a message digest	Requires 2^{128} operations to break in	Requires 2^{160} operations to break in, therefore more secure
Attack to try and find two messages producing the same message digest	Requires 2^{64} operations to break in	Requires 2^{80} operations to break in
Successful attacks so for	There have been reported attempts to some extent.	No such claims so far.
Speed	Faster (64 iterations and 128 bit buffer.)	Slower (80 iterations, and 160 bit buffer)
Software implementation	Simple, does not need any large programs or complex tables.	Simple does not need any large programs or complex table.

d. Usernames and passwords

A common method to provide user credentials during authentication is by providing a simple username and password to the server. Some solutions use the concept of a pre-shared key (PSK) instead, as it is more applicable to machines. One way to generate secure, difficult-to-guess usernames and passwords is to randomly create them. In this way, they correspond more to pre-shared keys. Both the server and the client need to be aware of this information. The identity must also be distributed among the entities that are to communicate with the device. In the case of XMPP, this problem has been solved. The XMPP Protocol. Here, the device creates its own random identity and creates the corresponding account in the XMPP server in a secure

manner. It then reports its identity to a Thing Registry or provisioning server where the owner can claim it and learn the newly created identity. This method never compromises the credentials and does not affect the cost of production negatively. Furthermore, passwords should never be stored in clear text. Important on servers where many passwords are stored. Instead, hashes of the passwords should be stored. Most modern authentication algorithms support the use of password hashes. Storing hashes minimizes the risk of unwanted generation of original passwords for attempted reuse in other systems.

e. Using message brokers and provisioning servers

Using message brokers can greatly enhance security in an IoT application and lower the complexity of implementation when it comes to authentication, if message brokers provide authenticated identity information in messages it forwards. In XMPP, all the federated XMPP servers authenticate clients connected to them as well as the federated servers themselves when they intercommunicate to transport messages between domains. This relieves clients from the burden of having to authenticate each entity in trying to communicate with it since they all have been securely authenticated. It's sufficient to manage security on an identity level. Even this step can be relieved further using provisioning, as described in Chapter 6, The XMPP Protocol. Unfortunately, not all protocols using message brokers provide this added security since they do not provide information about the sender of packets. MQTT is an example of such a protocol.

f. Centralization versus decentralization

When designing IoT architecture -avoid storing data in a central position if possible. Only store the data centrally which is actually needed to bind things together. Distribute logic, data, and workload. Perform work out of network as far as possible. This makes the solution more scalable, and it utilizes existing resources better. Use the linked data to spread data across the Internet, and use standardized grid computation technologies to assemble distributed data (for example, SPARQL) to avoid the need to store and replicate data centrally. Use a federated set of small local brokers instead of trying to get all the devices on the same broker.

7.7 THE NEED FOR INTEROPERABILITY

Interoperability is the ability of two or more devices, systems, platforms or networks to work in conjunction. Interoperability enables communication between heterogeneous devices or system in order to achieve a common goal. The devices and systems are fragmented with respect to the communication technologies, protocols, and data formats. This makes it difficult for the devices and systems in the IoT network to communicate and share their data with one another.

a. Solves complexity

Those companies that believe they can control the entire value chain, from things to services, middleware, administration, operation, apps. What will happen if they fail to operate. Companies that built devices with protocols, middleware, and mobile phone applications, where human can control things. Imagine a future where you have a thousand different things in your apartment from a hundred manufacturers. Would you want to download a hundred smart phone apps to control them? Would you like five different applications just to control your lights at home, just because you have light bulbs from five different manufacturers? An alternative would be to have one app to rule them based on requirement & feedback. To make it interoperable, they should communicate using a commonly understood language.

b. Reduces cost

Interoperability does not only affect simplicity of installation and management; it also takes care of the price of installation and solution to it. Companies that promote products, where you're forced to use their system to control your devices, can force their clients to pay a high price for future devices and maintenance, or the large investment made originally might be lost. Interoperability provides competition, and competition drives down cost and increases functionality and quality.

c. Allows new kinds of services and reuse of devices

New applications and services will be built that will reuse existing devices, which were installed perhaps as part of other systems and services. These applications will deliver new value to the inhabitants of the city without the need of installing new duplicate devices but such multiple use of devices is only possible if the devices communicate in an open and interoperable way. However, care must be taken at the same time since installing devices in an open environment requires the communication infrastructure to be secure as well. To build smart cities, it is important to use technologies that allow to have both a secure communication infrastructure and an interoperable.

d. Combining security and interoperability

Depending on the communication infrastructure, we might have to use security measures that directly oppose the idea of an interoperable infrastructure, prohibiting third parties from accessing existing devices in a secure fashion. It is important during the architecture design phase, before implementation, to thoroughly investigate what communication technologies are available, and what they provide and what they do not provide. All such implementation is by its very nature proprietary, and therefore not interoperable.

7.8 SUMMARY

In this we talked about the risks involved in IoT, Security and interoperability. There always a risk associated in any kind of work whether it is technical or non-technical. IoT security is the protection of Internet of Things devices from attack. While many business owners are aware that they need to protect computers and phones with antivirus, the security risks related to IoT devices are less well known and their protection is often neglected. While consumer IoT devices allow lifestyle benefits, businesses are quickly adopting IoT devices due to high potential for savings. IoT devices can greatly increase productivity for businesses, they also come with risks. Since IoT devices are connected to the internet, they can be hacked just like any other internet-enabled device. The attack surface of a network consists of all the possible places where it can be attacked, and it expands with every new internet-connected device. Even if the chance of one device being accessed by a perpetrator is small, the large number of IoT devices being brought into businesses can create a significant security risk.

7.9 REFERENCES FOR FUTURE READING

Learning Internet of Things by Peter Waher, Packt Publishing. (All notes are taken from this prescribed reference book).

Notes are taken from the link below:

https://blog.avast.com/iot-security-business-risk.

https://www.cisco.com/c/en_in/products/security/vpn-endpoint-security-clients/what-is-vpn.html.

https://www.imperva.com/learn/application-security/man-in-the-middle-attack-mitm/

7.10 UNIT END EXERCISE

1. What is a risk in IoT. Elaborate with an example.
2. What are the different modes of attacks in the network.
3. What do you mean by interoperability in IoT?
4. What is the need of interoperability in IoT.
5. Explain Denial of Service attack.
6. What do you mean by Man-in-the-Middle attack?
7. Describe in detail port scanning and Web Crawling?
8. How wildcards and breaking cipher helps in attacks in the network.
9. What are the different tools available to achieve security?
10. Explain Virtual Private Network (VNP)? What are its features.

❄❄❄❄❄❄❄

8

INTRODUCTION TO IOT

Unit Structure

8.1 OBJECTIVE

The aim of the Internet of Things to creation network is:

To coordinate and help to increase and optimize the utilization of results and value creation in IoT.

To create innovation strategies for the development of enabling technologies (nano-electronics, embedded systems, communication technologies, software, and cloud computing, etc.) required for IoT applications.

The main purpose of IoT is to create an ecosystem that connects everything. An ecosystem where everything is connected to each other is known as the Internet of Everything.

8.2 INTRODUCTION

Nowadays information technology is developed to be as one part that covers different phases related to the spread of the Internet and the Web into the tangible infrastructure which is called as the Internet of Things. The Internet of Things (IoT) is considered important knowledge that adapted the human to live in a smart and speed life, through enabling things to communicate with other objects, such as machines. IoT describes a system that contains many things that are exiting in the real world, along with sensors attached to them or embedded in these things, and connected to the Internet through a networked structure both as wireless or wired media.

8.3 DEFINITION

The IoT is «an interconnected environment in which all kinds of objects have a digital presence and the ability to communicate with other objects and people. IoT device comes with some common set of features like connectivity, analytics, endpoint management, etc.

The IoT enables the entities to connect and control the IoT devices present in the network by-

1. The entity can use a remote (tablets, smartphones) for sending a command or request for information over an IoT device.
2. The device then performs the command or can even send the information back over the network which has to be analyzed
3. This storing and analyzing of data can be carried out in multiple locations which include- cloud, local database or sometimes the data himself.

8.4 FEATURES OF INTERNET OF THINGS

Any IoT device should have the following features associated with it.

1. Connectivity

IoT devices can be connected over Radio waves, Bluetooth, Wi-Fi, Li-Fi, etc. Various protocols of internet connectivity layers can be used to maximize efficiency and establish connectivity across IoT Industry. Without seamless communication among the interrelated components of the IoT ecosystems (i.e sensors, compute engines, data hubs, etc.) it is not possible to execute any business.

2. Sensing

In the case of IoT we need to read the analog signal, convert it in such a way that we can derive meaningful insights out of it. Use of electrochemical, gyroscope, pressure, light sensors, GPS, Electrochemical, pressure, RFID, etc. to gather data based on an analysis. For example, for automotive use cases, we use Light detection sensors along with pressure, velocity and imagery sensors.

3. Active Engagements

IoT device connects various products, technologies and services work together by establishing an active engagement between them. Cloud computing is used to establish active engagements among IoT components. In the case of Industry grade, IoT takes the raw data which need to be acquired, preprocessed, and rescale as per business capacity. While designing an IoT ecosystems carriers need to consider the future needs of manipulating such a huge scale of data to satisfy incremental business needs.

4. Scale

IoT devices should be designed in such a way that they can be scaled up or down easily on demand based on market or industry standards. In general, IoT is being used from smart home automation to automating large factories and workstations. A carrier should design their IoT infrastructure depending upon their current and future engagement scale.

5. Dynamic Nature

For any IoT use case, the first and important step is to collect and convert data in such a way that means business decisions can be made of it. In this whole process, various components of IoT need to change their state dynamically. For example, the input of a temperature sensor will vary continuously based on weather conditions, locations, etc. IoT devices should be designed this keeping in mind.

6. Intelligence

In IoT, the data is used to make important business insights and drive important business decisions based on the analysis. Various machine learning & deep learning algorithms are built to models on top of this massive data to obtain valuable insights. The analog signals are preprocessed and converted to a format on which machine-learning models are trained.

7. Energy

From end components to connectivity and analytics layers, the whole ecosystems demand a lot of energy. While designing an IoT ecosystem, one need to consider design methodology such that it should be eco-friendly and energy consumption is minimal moreover recycling can also be done.

8. Safety

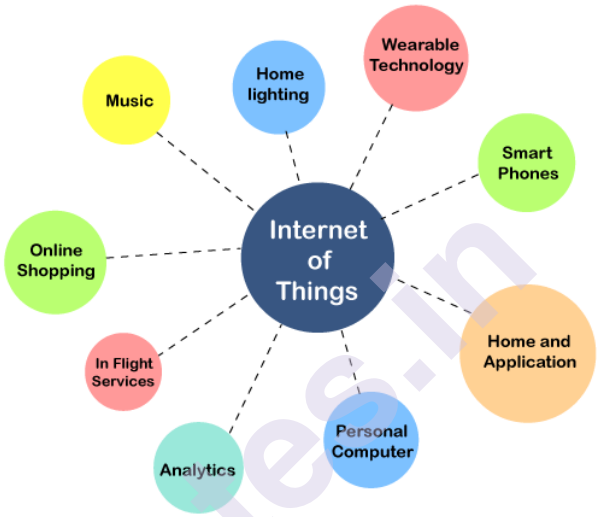
One of the main features of the IoT ecosystem is security. In the network of an IoT system, crucial and sensitive information travels from endpoints to the analytics layer via connectivity components. While designing an IoT system we need to adhere to proper safety, security measures, and firewalls to keep the data away from misuse and manipulations. Compromising any component of an IoT system can eventually lead to failure of the whole system.

9. Integration

IoT integrates various cross-domain models to enrich user experience. It also ensures proper trade-off between infrastructure and operational costs.

8.5 APPLICATIONS OF IOT

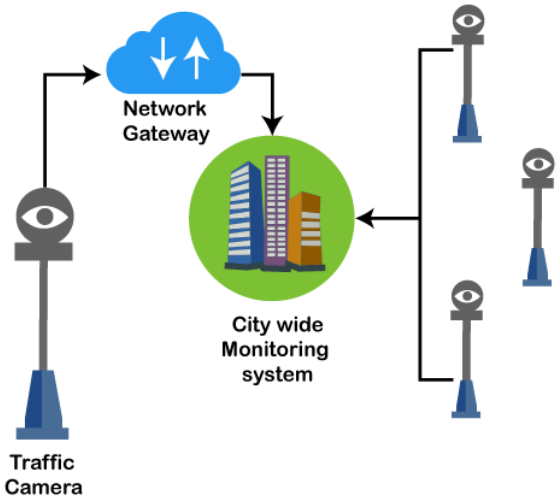
The Internet of Things (IoT) provides the ability to interconnect computing devices, mechanical machines, objects, animals or unique identifiers and people to transfer data across a network without the need for human-to-human or human-to-computer is a system of conversation. IoT applications bring a lot of value in our lives. The Internet of Things provides objects, computing devices, or unique identifiers and people's ability to transfer data across a network without the human-to-human or human-to-computer interaction.



Source : https://www.javatpoint.com/internet-of-things-applications

A traffic camera is an intelligent device. The camera monitors **traffic congestion, accidents** and **weather conditions** and can access it to a common entrance.

This gateway receives data from such cameras and transmits information to the city's **traffic monitoring system**.



Source: https://www.javatpoint.com/internet-of-things-applications

1. Wearables

Wearable technology is the hallmark of IoT applications and one of the earliest industries to deploy IoT. Devices are designed for heart rate monitors and smartwatches etc. nowadays.

Guardian glucose monitoring device has been developed to help people with diabetes. It detects glucose levels in our body, uses a small electrode called the glucose sensor under the skin, and relates it to a radiofrequency monitoring device.

2. Smart Home Applications

The smart home or smart city talks about the IoT application. AI home automation is employed, home automation system, where a string of musical notes uses in-house functions.

3. Health care

IoT applications can transform reactive medical-based systems into active wellness-based systems. Resources uses controlled environments, leftover data, and volunteers for clinical trials. The Internet of Things improves the device's power, precision, and availability.

4. Smart Cities

Smart city uses technology of IoT to provide services. The smart city includes improving transportation and social services, promoting stability etc.

Engineers use the Internet of Things to analyze the complex factors of town and each city. IoT applications help in water management, waste control and emergencies.

Eg: Smart city - Palo Alto.

Palo Alto, San Francisco, is the city to acquire the traffic approach. They realized that most cars roam around the same block on the streets in search of parking spots. It is the primary cause of traffic congestion in the city. Thus, the sensors were installed at all parking areas in the city. These sensors pass occupancy status to the cloud of each spot. And hence the vacant space for cars is identified and cars can be parked properly.

5. Agriculture

To feed such a large population, agriculture needs to collaborate with technology and get the best results out of it. Eg: Smart Greenhouse where farming techniques grow crops by environmental parameters. Traditional method of manual handling results in production losses, energy losses and labor costs, making it less effective and involves huge amount of loss due to weather conditions. The greenhouse makes it easy to monitor and enables to control the climate inside it.

6. Industrial Automation

In industry automation is must where the quality of products is an essential factor for a more significant investment return and good turnover. Anyone can design or re-construct products and their packaging to provide superior quality and performance in cost and customer experience with IoT applications. IoT will prove as a game-changer. In industrial automation, IoT is used in the areas such as:

- Product flow monitoring
- Factory digitization
- Inventory management
- Safety and security
- Logistics and Supply Chain Optimization
- Quality control
- Packaging customization

7. Hacked Car

A connected car is a technology-driven car with Internet access and a WAN network. The technology offers the user some benefits such as in-car infotainment, advanced navigation and fuel efficiency.

8. Healthcare

Healthcare does real-time monitoring with the help of smart devices. It gathers and transfers health data such as blood pressure, blood sugar levels, weight, oxygen, and ECG. The patient can contact the doctor by the smart mobile application in case of any emergency.

9. Smart Retail

IoT applications in retail give shoppers a new experience. Customers do not have to stand in long queues as the checkout system can read the tags of the products and deduct the total amount from the customer's payment app with IoT applications' help.

10. Smart Supply Chain

Customers automate the delivery and shipping with a smart supply chain. It also provides details of real-time conditions and supply networks.

11. Smart Farming

Farmers can minimize waste and increase productivity. The system allows the monitoring of fields with the help of sensors. Farmers can monitor the status of the area.

8.6 IOT EXAMPLES IN REAL LIFE

1. Sensors

IoT sensors consist of manual or digital sensors connected to circuit boards such as Arduino Uno or Raspberry Pi 2. The circuit boards can be programmed to measure a range of data collected from a sensor device such as carbon monoxide, temperature, humidity, pressure, vibration, and motion. IoT sensors gather data at different physical environments and sends data to the connected devices. They can be used by businesses for predictive maintenance, enhanced efficiency, and reduced costs.

2. Data Analysis

Businesses are increasingly using IoT data analytics to determine trends and patterns by analyzing big and small data. IoT data analytics apps can analyze structured, unstructured, and semi-structured data to extract meaningful insights and predict the result.

IoT can be applied to data analytics to investigate different types of data including motion data sets, geographical data, and health care data. It can be used by businesses for predictive and descriptive analysis to improve customer knowledge, enhance operational efficiency, and create business value.

3. Tracking and Monitoring Systems

A lot of industry such as Amazon, Flip Kart etc. are using IoT systems for asset tracking. IoT asset tracking devices use GPS or radio frequency (RF) to track and monitor the product. The smart devices can be used for long-range identification and verification of assets.

4. Smart Supply Chain Management

Supply chain managers can make improved predictions through smart routing and rerouting algorithms. IoT devices are tagged to packages that can provide instant location finding facility via GPS and RFID signals that can help to make informed supply chain decisions. IoT applications can help in mitigating uncertainty risks in supply chain management. Supply chain managers can make use of smart supply chain management programs for minimizing variance, reducing costs, and improving profitability. The programs can help in inventory management, vendor relationship, fleet management, and scheduled maintenance.

5. Smart Barcode Readers

IoT barcode readers can help in better inventory management for retailers. The readers support AI-based digital signal processing. These devices can optimize the operations of many sectors including retail, logistics, warehouse, and much more.

IoT based bar card readers feature cloud data connections to connect with other systems. Using the connected bar code reader will ease the process of managing inventory.

IoT barcode readers can be incorporated into shopping carts. The readers use AI-based sensor to detect products as they are dropped or removed from the cart. The reader can transfer data to the computer automatically, and that can save a lot of time in checkout resulting in an improved experience for the customers.

6. Smart Grids

The smart grid is another industrial application of IoT. The grid allows real-time monitoring of data regarding supply and demand of electricity. It involves the application of computer intelligence for the efficient management of resources.

Utility companies can use IoT smart grid technologies for more efficient outage management. They can use the technology to identify load distribution and improve reliability. The technology can also assist in fault detection and repairs.

With the smart grid, utilities can interconnect all their assets including meters and substations. Applying IoT technologies to the grid ecosystem allows utility companies to exercise greater control over the power infrastructure and resources. Moreover, they allow consumers with better quality access to energy.

7. Connected HealthCare System

IoT has numerous applications in the healthcare industry. The technology can be used to provide high-quality medical services using smart medical devices.

IoT automates the workflow by allowing the provision of effective health care services to the patients.

IoT medical devices can help in real-time monitoring of patients remotely. The devices can report an emergency like an asthma attack, heart failure, etc., immediately to a physician. This can help in potentially saving the lives of many individuals.

IoT devices can collect health care data including blood pressure, sugar levels, oxygen, and weight. Data is stored online and can be accessed anytime by a physician.

8. Smart Farming

Farmers can use smart IoT farming applications for optimizing a lot of different activities such as determining the best time to harvest plants, creating fertilizer based on the chemicals in the soil, and sensing soil nutrients and moisture levels.

IoT technologies can help in precise farming which can result in optimized production. Some IoT devices and sensors can detect weather conditions and other environmental data. Applications of IoT technologies can help to boost both the quality and quantity of agriculture production.

8.7 SIMPLE IOT LED PROGRAM

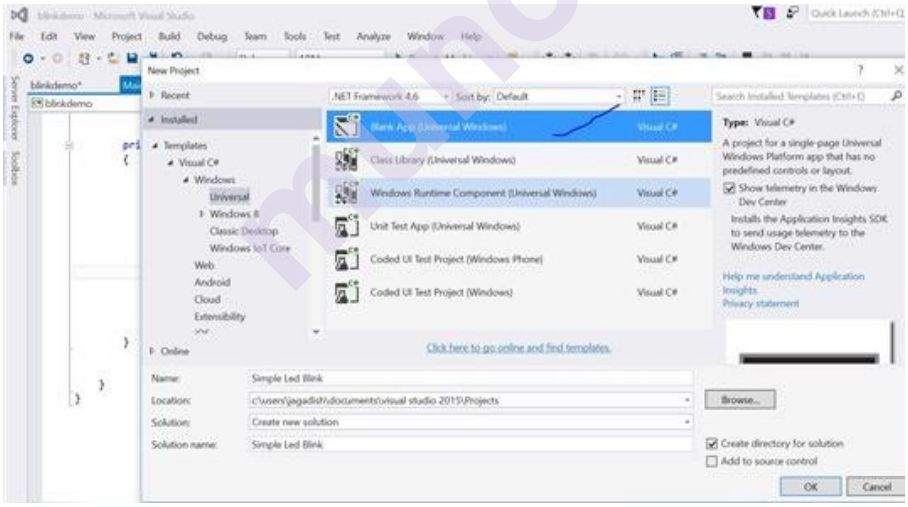
Simple LED Blink example

Aim: Simple experiment of connecting LED lights with Raspberry Pi device (device only with Windows 10 IoT Core OS)

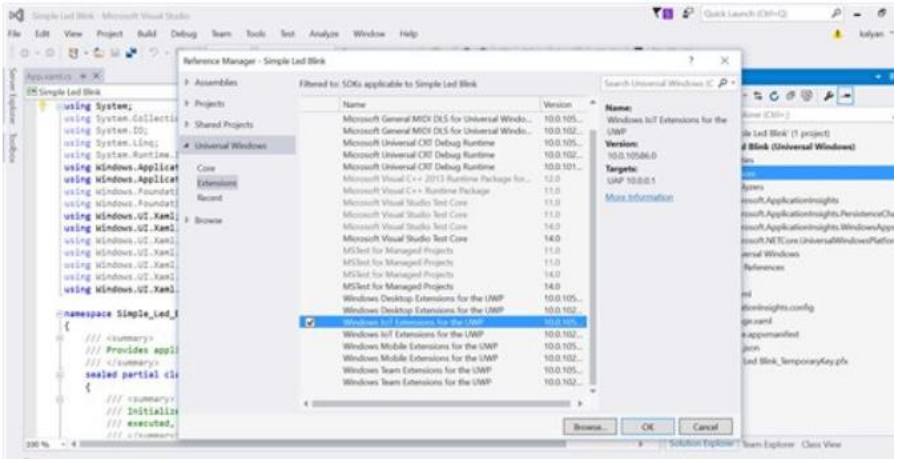
Requirements:

- Raspberry Pi 2/3 with Windows 10 IoT Core OS connected to internet
- Laptop or PC with Windows 10 OS connected to internet
- Visual Studio 2015 installed on your laptop or PC
- Bread Board
- LED Light
- Resistor

Step – 01: Select File -> Create new project -> Name it "Simple Led Blink"



Step – 02: Right click on the selection and add reference. Select Windows IoT Extension for the UWP and click ok.



Step – 03: Open MainPage.xaml and add the following for UI

```
<Grid Background="Wheat">

   <StackPanel HorizontalAlignment="Center"
VerticalAlignment="Center">

      <Ellipse x:Name="LED" Fill="LightGray" Stroke="White"
Width="100" Height="100" Margin="10"/>

      <TextBlock x:Name="DelayText" Text="500ms" Margin="10"
TextAlignment="Center" FontSize="26.667" />

      <TextBlock x:Name="GpioStatus" Text="Waiting to initialize
GPIO..." Margin="10,50,10,10" TextAlignment="Center"
FontSize="26.667" />

   </StackPanel>

 </Grid>
```

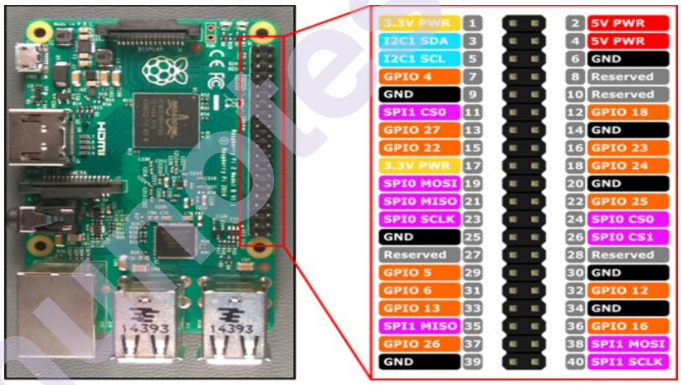
Step – 04: Open the MainPage.xaml.cs add the following in the cs file.

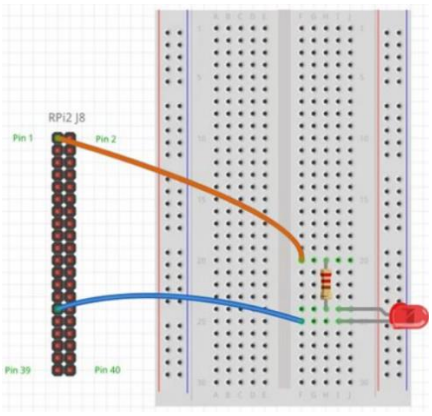
```
private const int LED_PIN =5;

 private GpioPin pin;

 private GpioPinValue pinValue;

 private DispatcherTimer timer;

 private SolidColorBrush redBrush = new
SolidColorBrush(Windows.UI.Colors.Red);

 private SolidColorBrush grayBrush = new
SolidColorBrush(Windows.UI.Colors.LightGray);

  public MainPage()

 {

   InitializeComponent();
```

```
  timer = new DispatcherTimer();
  timer.Interval = TimeSpan.FromMilliseconds(500);
  timer.Tick += Timer_Tick;
  InitGPIO();
  if (pin != null)
  {
    timer.Start();
  }
}
private void InitGPIO()
{
  var gpio = GpioController.GetDefault();

  // Show an error if there is no GPIO controller
  if (gpio == null)
  {
    pin = null;
    GpioStatus.Text = "There is no GPIO controller on this device.";
    return;
  }
  pin = gpio.OpenPin(LED_PIN);
  pinValue = GpioPinValue.High;
  pin.Write(pinValue);
  pin.SetDriveMode(GpioPinDriveMode.Output);
   GpioStatus.Text = "GPIO pin initialized correctly.";
}
private void Timer_Tick(object sender, object e)
{
  if (pinValue == GpioPinValue.High)
```

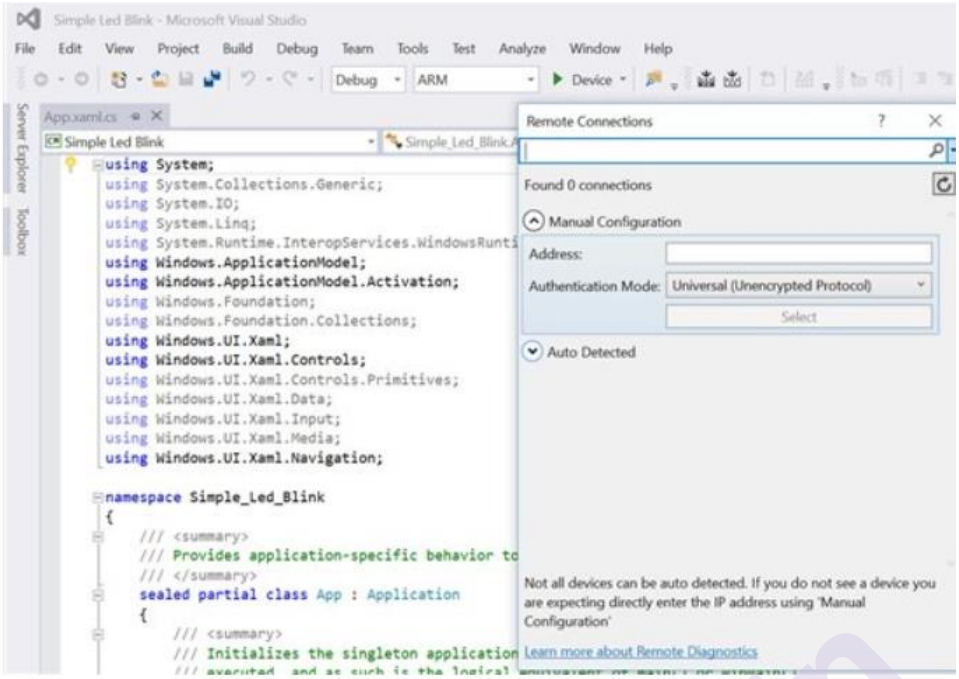
```
        {
          pinValue = GpioPinValue.Low;
          pin.Write(pinValue);
          LED.Fill = redBrush;
        }
        else
        {
          pinValue = GpioPinValue.High;
          pin.Write(pinValue);
          LED.Fill = grayBrush;
        }
      }

}
```

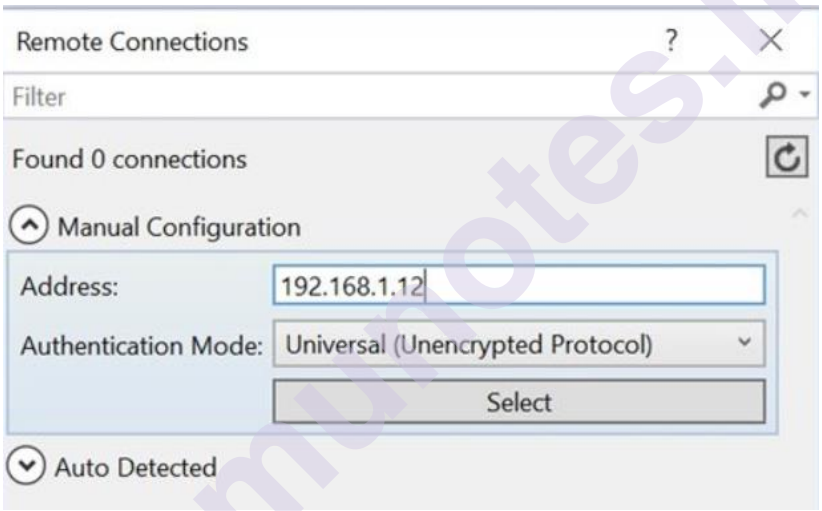
Use GPIO (General Purpose Input Output) 5 over here





Step – 05: Now build the solution and select ARM and select Remote Machine, In the address column enter the Ip address of the network connected with Raspberry Pi device and click ok.





Step – 06: Now once Remote Machine is selected build and run the application. Once the application is build & run operation is completed successfully in the sense you will find the LED blinking according to the timer set.

Source:
https://social.technet.microsoft.com/wiki/contents/articles/33948.iot-simple-led-blink-example.aspx

8.8.SUMMARY

The Internet of Things (IoT) provides the ability to interconnect computing devices, mechanical machines, objects, animals or unique identifiers and people to transfer data across a network without the need for human-to-human or human-to-computer is a system of conversation.

IoT applications bring a lot of value in our lives. The Internet of Things provides objects, computing devices, or unique identifiers and people's ability to transfer data across a network without the human-to-human or human-to-computer interaction.

IoT is used in real life applications such as –health monitoring systems, whether forecasting, tracking of the product, inventory management, smart grid and many more.

8.9.REFERENCES FOR FUTURE READING

Notes taken from the link below:

https://www.javatpoint.com/internet-of-things-applications

https://www.educba.com/iot-features/

https://www.softwaretestinghelp.com/best-iot-examples/#10_Best_Real-World_IoT_Examples

https://social.technet.microsoft.com/wiki/contents/articles/33948.iot-simple-led-blink-example.aspx

8.10 UNIT END EXERCISE

What is IoT?

What are the features of IoT?

Explain the applications of IoT

With an example explain the real-life application of IoT?

Elaborate the process of IoT in supply chain management system?

Explain how IoT is applicable to medical industry?

Write a note on “Improvement in traffic congestion relief using IoT?

❄❄❄❄❄❄❄