

Prof: Siddhesh Zele's



PHYSICAL COMPUTING AND IOT PROGRAMMING

SYBSC-CS

SEM 3

*302 PARANJPE UDYOG BHAVAN, NEAR KHANDELWAL SWEETS, NEAR THANE
STATION, THANE (WEST)*

PHONE NO: 8097071144 / 8097071155

INDEX

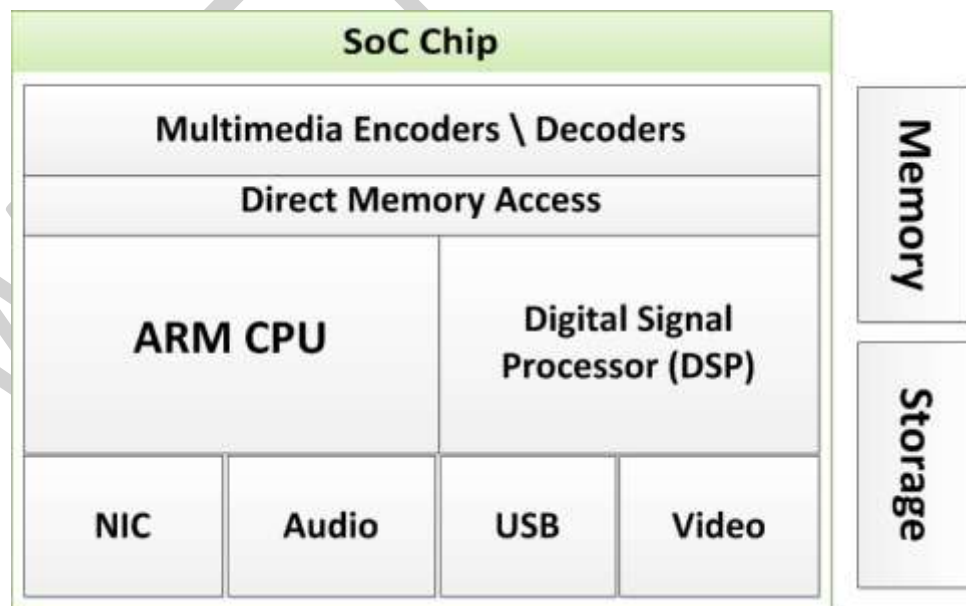
UNIT	CONTENT	PAGE NO
Unit I	<p>SoC and Raspberry Pi</p> <p>System on Chip: What is System on chip? Structure of System on Chip.</p> <p>SoC products: FPGA, GPU, APU, Compute Units.</p> <p>ARM 8 Architecture: SoC on ARM 8. ARM 8 Architecture Introduction</p> <p>Introduction to Raspberry Pi: Introduction to Raspberry Pi, Raspberry Pi Hardware, Preparing your raspberry Pi.</p> <p>Raspberry Pi Boot: Learn how this small SoC boots without BIOS. Configuring boot sequences and hardware.</p>	1
Unit II	<p>Programming Raspberry Pi</p> <p>Raspberry Pi and Linux: About Raspbian, Linux Commands, Configuring Raspberry Pi with Linux Commands</p> <p>Programing interfaces: Introduction to Node.js, Python.</p> <p>Raspberry Pi Interfaces: UART, GPIO, I2C, SPI Useful Implementations: Cross Compilation, Pulse Width Modulation, SPI for Camera.</p>	12
Unit III	<p>Introduction to IoT: What is IoT? IoT examples, Simple IoT LED Program.</p> <p>IoT and Protocols IoT Security: HTTP, UPnp, CoAP, MQTT, XMPP.</p> <p>IoT Service as a Platform: Clayster, Thinger.io, SenseIoT, carriots and Node RED.</p> <p>IoT Security and Interoperability: Risks, Modes of Attacks, Tools for Security and Interoperability.</p>	31

UNIT 1

System on a chip

- A system on a chip or system on chip (SoC or SOC) is an integrated circuit (also known as an "IC" or "chip") that integrates all components of a computer or other electronic systems.
- It may contain digital, analog, mixed-signal, and often radio-frequency functions—all on a single substrate.
- SoCs are very common in the mobile computing market because of their low power-consumption.
- typical application is in the area of embedded systems.
- The contrast with a microcontroller, SoC integrates microcontroller (or microprocessor) with advanced peripherals like graphics processing unit (GPU), Wi-Fi module, or coprocessor.
- **In general, we can distinguish three types of SoC**
 1. SoC built around a microcontroller
 2. SoC built around a microprocessor (this type can be found in mobile phones)
 3. specialized SoC designed for specific applications that do not fit into the above two categories.

Structure/Architecture of SOC



A typical SoC consists of:

1. a microcontroller, microprocessor or digital signal processor (DSP) core – multiprocessor SoCs (MPSoC) having more than one processor core
2. memory blocks including a selection of ROM, RAM, EEPROM and flash memory
3. timing sources including oscillators and phase-locked loops
4. peripherals including counter-timers, real-time timers and power-on reset generators
5. external interfaces, including industry standards such as USB, FireWire, Ethernet, USART, SPI
6. analog interfaces including ADCs and DACs
7. voltage regulators and power management circuits

FPGA

Processors and FPGAs (field-programmable gate arrays) are the hardworking cores of most embedded systems. Integrating the high-level management functionality of processors and the stringent, real-time operations, extreme data processing, or interface functions of an FPGA (Field Programmable Gate Array) into a single device forms an even more powerful embedded computing platform.

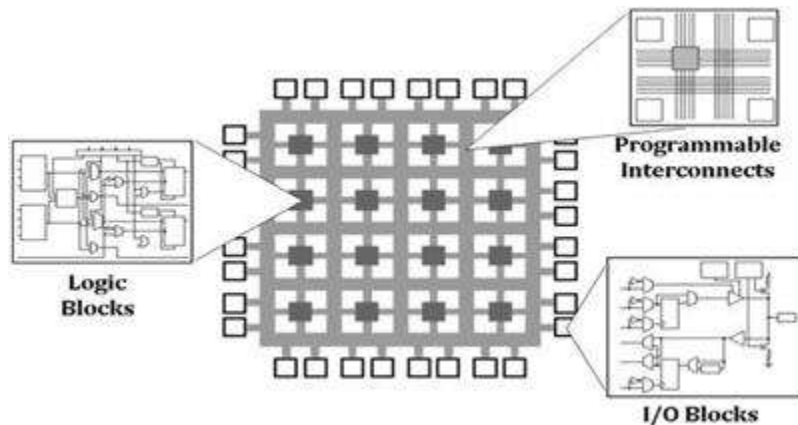
SoC FPGA devices integrate both processor and FPGA architectures into a single device. Consequently, they provide higher integration, lower power, smaller board size, and higher bandwidth communication between the processor and FPGA. They also include a rich set of peripherals, on-chip memory, an FPGA-style logic array, and high speed transceivers.

A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence "field-programmable".

FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together", like many logic gates that can be inter-wired in different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR.

In most FPGAs, logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory

An FPGA can be used to solve any problem which is computable. This is trivially proven by the fact FPGA can be used to implement a soft microprocessor, such as the Xilinx MicroBlaze or Altera Nios II. Their advantage lies in that they are sometimes significantly faster for some applications because of their parallel nature and optimality in terms of the number of gates used for a certain process.



FPGA Advantages

1. **Long time availability** FPGAs (Field Programmable Gate Arrays) enable you to make yourself independent from component manufacturers and distributors since the functionality is not given by the device itself but in its configuration.
2. **Can be updated and upgraded at your customer's site** FPGAs in contrast to traditional computer chips are completely configurable.
3. **Fast and efficient systems** Available standard components address a broad user group and consequently often constitute a compromise between performance and compatibility.
4. **Performance gain for software applications** Complex tasks are often handled through software implementations in combination with high-performance processors.
5. **Real time applications** FPGAs are perfectly suitable for applications in time-critical systems. In contrast to software based solutions with real time operating systems, FPGAs provide real deterministic behavior.

Graphics processing unit

- A graphics processing unit (GPU), occasionally called visual processing unit (VPU), is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device.
- GPUs are used in embedded systems, mobile phones, personal computers, workstations, and game consoles.
- Modern GPUs are very efficient at manipulating computer graphics and image processing, and their highly parallel structure makes them more efficient than general-purpose CPUs for algorithms where the processing of large blocks of data is done in parallel.

In personal computers, there are two main forms of GPUs.

- Dedicated graphics card - also called discrete.
- Integrated graphics - also called: shared graphics solutions, integrated graphics processors (IGP), or unified memory architecture (UMA).

APU

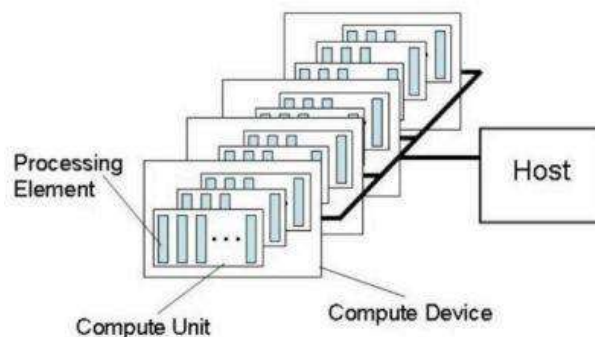
- An Acceleration Processing Unit (APU) is a coprocessor that may be integrated with CPU on a common chip or within the same package, interconnected via a standard processor bus. This approach offers low latency interactive access operations without a modification to the processor design layout.
- An APU may also be used to accelerate the processing of a specialized intense computation engine of a stable purpose that justifies the cost of integration and does not require reconfiguration.
- A system on a chip or system on chip (SoC) integrates a complex system components on a single chip. It typically includes one or more processors (or DSP cores), memory, and I/O interfaces; and is employed in data processing, communications, and control environments. A SoC may be implemented using different technologies, including FPGAs.
- The AMD Accelerated Processing Unit (APU), formerly known as Fusion, is the marketing term for a series of 64-bit microprocessors from AMD designed to act as a CPU and graphics accelerator (GPU) on a single chip.

COMPUTE UNITS OF GPU

The meaning of a compute unit depends on who manufactured a particular GPU

- A compute unit is a stream multiprocessor in a NVidia GPU
- SIMD engine in an AMD GPU

Each compute unit has several processing elements (ALU/stream processor)



One compute unit combines 64 shader (In the field of computer graphics, a shader is a computer program that is used to do shading: the production of appropriate levels of light, darkness, and color within an image, or, in the modern era, also to produce special effects or do video post-processing.) processors with 4 TMUs (A texture mapping unit (TMU) is a component in modern graphics processing units (GPUs), historically it was a separate physical processor. A TMU is able to rotate, resize, and distort a bitmap image (performing texture sampling), to be placed onto an arbitrary plane of a given 3D model as a texture.).

ARM 8 ARCHITECTURE

Introduction

- ARM, originally Acorn RISC Machine, later Advanced RISC Machine, is a family of reduced instruction set computing (RISC) architectures for computer processors, configured for various environments.
- British company ARM Holdings develops the architecture and licenses it to other companies, **who design their own products that implement one of those architectures—including systems-on-chips (SoC) and systems-on-modules (SoM) that incorporate memory, interfaces, radios, etc.**
- It also designs cores that implement this instruction set and licenses these designs to a number of companies that incorporate those core designs into their own products.
- Processors that have a RISC architecture typically require fewer transistors than those with a complex instruction set computing (CISC) architecture (such as the x86 processors found in most personal computers), which improves cost, power consumption, and heat dissipation.
- These characteristics are desirable for light, portable, battery-powered devices—including smartphones, laptops and tablet computers, and other embedded systems.

ARMv8-A Architecture

The ARMv8 architecture introduces 64-bit support to the ARM architecture with a focus on power-efficient implementation while maintaining compatibility with existing 32-bit software.

By adopting a clean approach ARMv8-A processors extend the performance range available while maintaining the low power consumption characteristics of the ARM processors that will power tomorrow's most innovative and efficient devices.

ARM has 3 different product tiers supporting the ARMv8-A architecture:

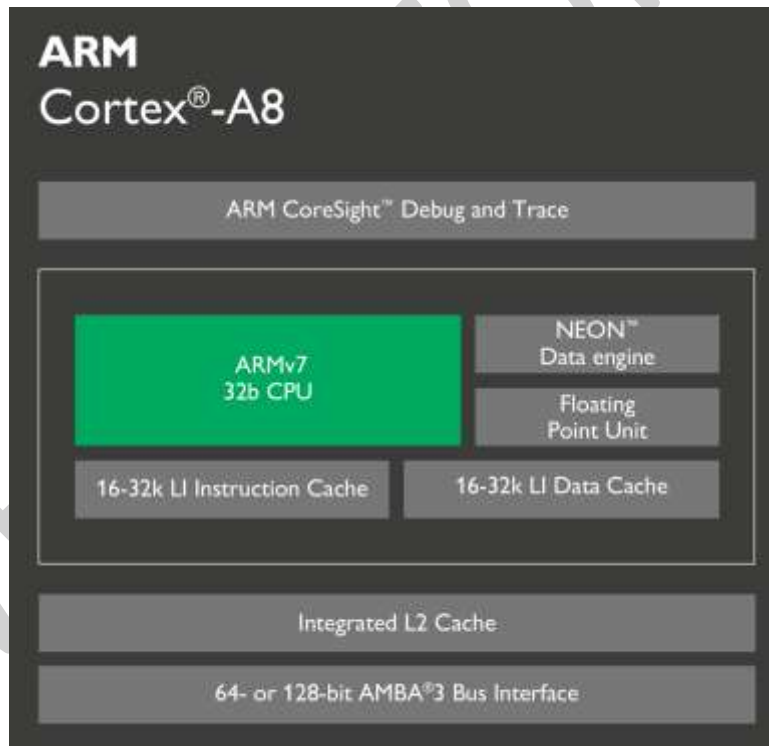
1. High Performance,
2. High Efficiency, and
3. Ultra-High Efficiency.

ADDRESS: 302 PARANJPE UDYOG BHVAN, ABOVE KHANDELWAL SWEETS, THANE STATION ROAD(W)
PHONE : 8097071144/8097071155

Increased availability of larger registers for general purpose and media instructions, a greater addressing range and cryptography instructions enable new categories of applications for superphone and tablet computing, while bringing the ARM benefits of efficient design and low power consumption to applications where 64-bit computing is already established, such as servers and network infrastructure, promising to revolutionize the data center.

The ARMv8 architecture maintains compatibility with the comprehensive software ecosystem for 32-bit components. This enables a wealth of software optimized for existing ARM processors to benefit from the enhanced performance of processors based on the ARMv8 architecture, while the addition of 32-bit cryptographic instructions further enables optimization for emerging requirements.

Developing the software to make best use of the new 64-bit capabilities requires the availability of excellent tools, test platforms and key open source components. While developing the architecture and the processors based on ARMv8-A, ARM has also ensured that the essential tools for development are available to software developers today, enabling the ARM software ecosystem to continue to innovate around the Architecture for the Digital World.



Introduction to Raspberry pi

- The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries.
- The original model became far more popular than anticipated selling outside of its target market for uses such as robotics.
- The first generation (Raspberry Pi 1 Model B) was released in February 2012. It was followed by a simpler and inexpensive model Model A
- the foundation released a board with an improved design in Raspberry Pi 1 Model B+. These boards are approximately credit-card sized and represent the standard mainline form-factor.
- Improved A+ and B+ models were released a year later. A "compute module" was released in April 2014 for embedded applications, and a Raspberry Pi Zero with smaller size and reduced input/output (I/O) and general-purpose input/output (GPIO) capabilities was released in November 2015
- The Raspberry Pi 2 which added more RAM was released in February 2015. Raspberry Pi 3 Model B released in February 2016, is bundled with on-board WiFi, Bluetooth and USB boot capabilities
- All models feature a Broadcom system on a chip (SoC), which includes an ARM compatible central processing unit (CPU) and an on-chip graphics processing unit (GPU, a VideoCore IV).
- CPU speed ranges from 700 MHz to 1.2 GHz for the Pi 3 and on board memory range from 256 MB to 1 GB RAM. Secure Digital(SD) cards are used to store the operating system and program memory in either the SDHC or MicroSDHC sizes.
- Most boards have between one and four USB slots, HDMI and composite video output, and a 3.5 mm phono jack for audio.
- the Pi 3 and Pi Zero W have on board Wi-Fi 802.11n and Bluetooth.

Raspberry Pi Hardware

Hardware	Features	Comments
System on a chip	Broadcom BCM2835	CPU, GPU, DSP, SDRAM, and USB port
CPU model	ARM1176JZF-S core	With floating point
Clock rate	700 MHz	Overclockable to 800 MHz
GPU	Broadcom VideoCore IV	
	OpenGL ES 2.0	3D
	OpenVG	3D
	MPEG-2	
	VC-1	Microsoft, licensed
	1080p30 H.264	Blu-ray Disc capable, 40 Mbit/s

	MPEG-4	AVC high-profile decoder and encoder
	1 Gpixel/s, 1.5 Gtexels/s	24 GFLOPS with DMA
Video output	Composite RCA	PAL and NTSC
	HDMI	Rev 1.3 and 1.4
	Raw LCD panels	Via DSI
Audio output	3.5 mm jack	
	HDMI	
Storage	SD/MMC/SDIO	Card slot
Peripherals	8 × GPIO	
	UART	
	I2C bus	100 kHz
	SPI bus	Two chip selects, +3.3 V, +5 V, ground
Power source	5 V via micro-USB	

Preparing Yours raspberry Pi

Installing raspberry pi

Before you start, you (obviously) should have a microSD card and a computer with an SD card reader. Besides that, download the Raspbian image file directly from raspberrypi.org (don't forget to unzip the image file!). Done? Good, let's get started.

Image : <https://www.raspberrypi.org/downloads/raspbian/>
select : RASPBIAN JESSIE WITH PIXEL

1. Insert your microSD card into your card reader and find out its drive letter in Windows Explorer (for example G:).
2. Download Win32DiskImager, unzip the downloaded file and run the utility file.
3. Select the Raspbian image file you downloaded.
4. Select the drive of your SD card in the 'Device' dropdown. Make sure you chose the correct one. Otherwise, you risk damaging the data on your hard drive.
5. Select 'Write' and wait for the process to finish. That's it!
6. Now you can plug the SD card into your Raspberry Pi's slot.

How to connect to Raspbian Os

Head mode:

1. led tv or monitor with hdmi cable
2. keyboard
3. mouse

ADDRESS: 302 PARANJPE UDYOG BHVAN, ABOVE KHANDELWAL SWEETS, THANE STATION ROAD(W)
PHONE : 8097071144/8097071155

4. lan cable
5. 5volt 5Amp power connector

Headless mode:

1. laptop or pc connected to lan or wifi (with telnet client program)
2. raspberrypi connected to same network (telnet server enabled) default
3. if DHCP is configured raspberrypi will get ipaddress to come in network
4. from laptop connect that ip address using putty
5. username : pi & password : raspberry

How this small Soc boots without BIOS

- The firmware is closed-source proprietary code programmed into the SoC (System on a Chip) processor, which cannot be modified.
 - Upon power-up the firmware will initiate a bootloader on the SD card
 - After this point everything else comes from the SD card.
1. First stage bootloader - This is used to mount the FAT32 boot partition on the SD card so that the second stage bootloader can be accessed. It is programmed into the SoC itself during manufacture of the RPi and cannot be reprogrammed by a user.
 2. Second stage bootloader (bootcode.bin) - This is used to retrieve the GPU firmware from the SD card, program the firmware, then start the GPU.
 3. GPU firmware (start.elf) - Once loaded, this allows the GPU to start up the CPU. An additional file, fixup.dat, is used to configure the SDRAM partition between the GPU and the CPU. At this point, the CPU is release from reset and execution is transferred over.
 4. User code - This can be one of any number of binaries. By default, it is the Linux kernel (usually named kernel.img), but it can also be another bootloader (e.g. U-Boot), or a bare-bones application.

Configuring booting sequence and hardware

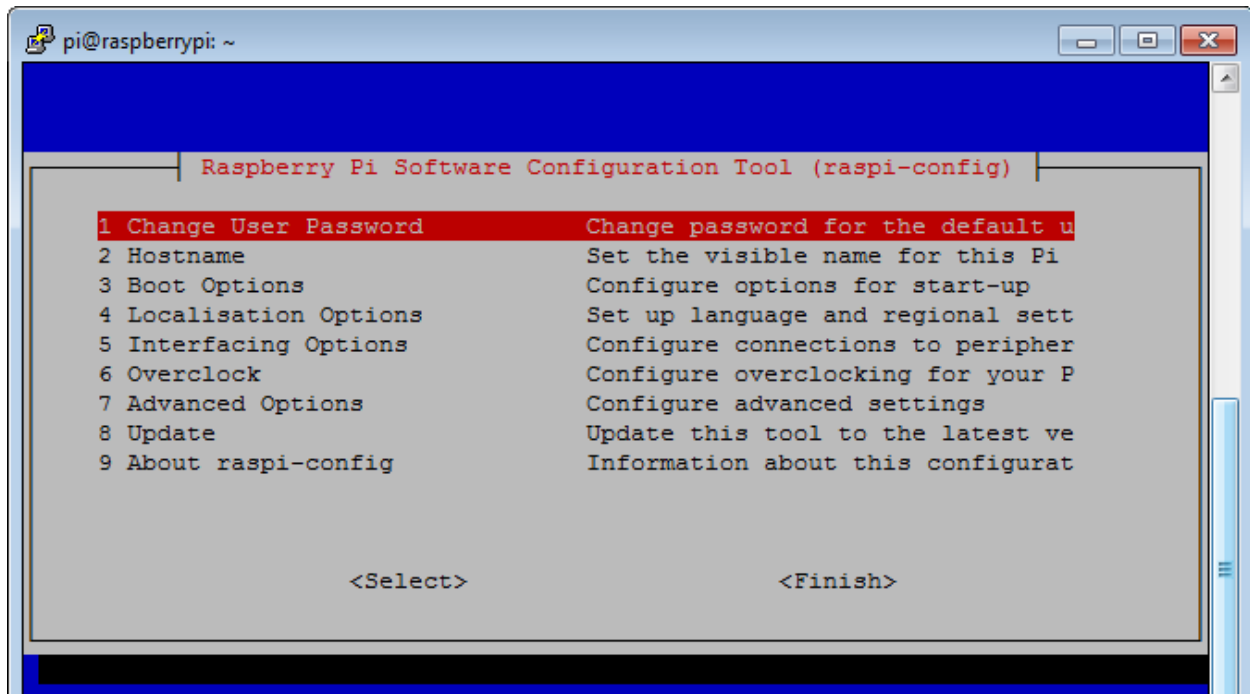
raspi-config is the Raspberry Pi configuration tool written and maintained by Alex Bradbury. It targets Raspbian.

sudo raspi-config

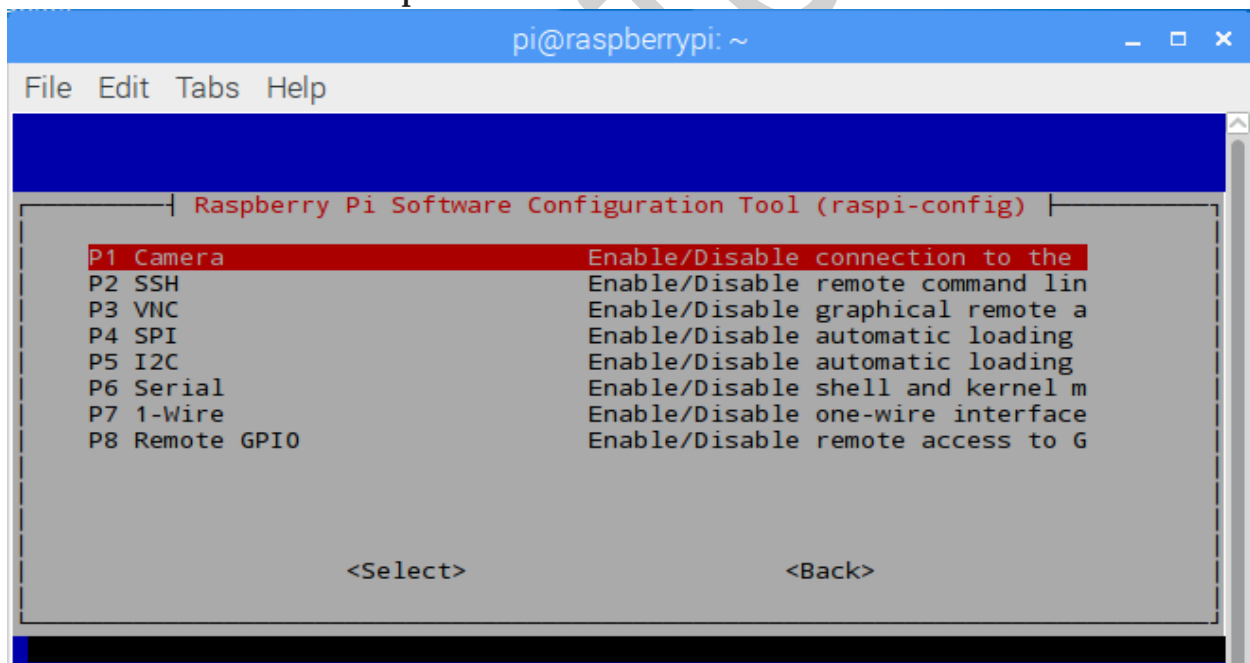
The sudo is required because you will be changing files that you do not own as the pi user.

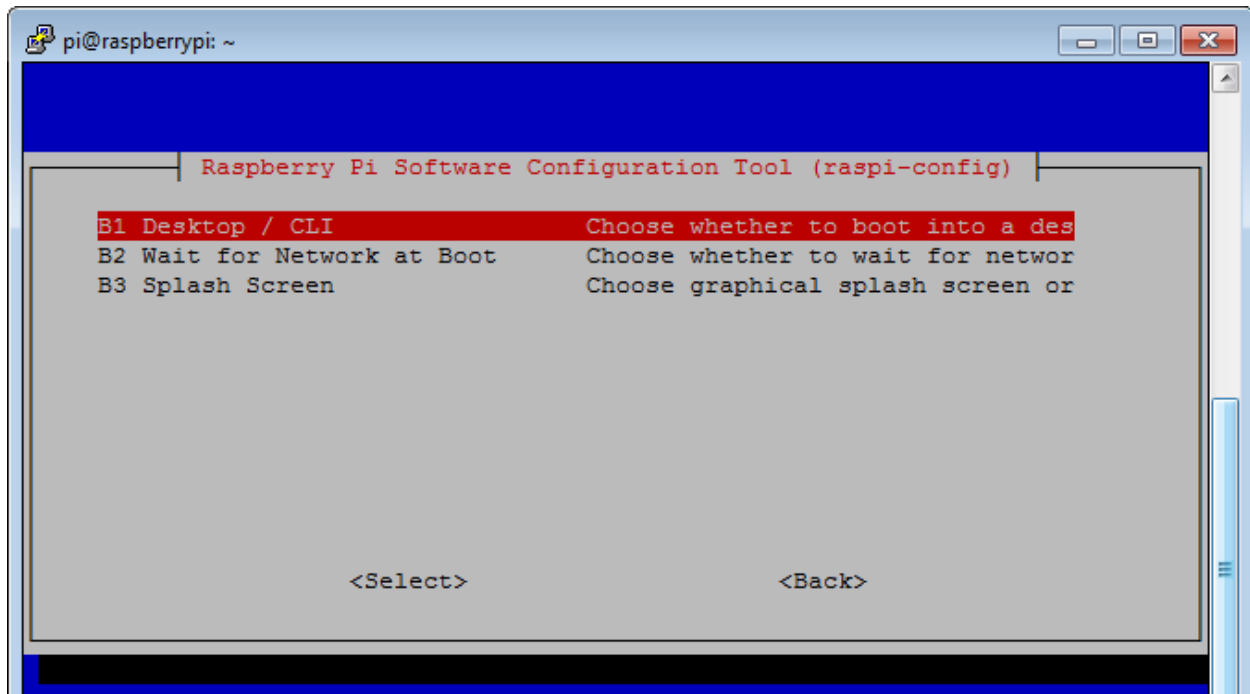
You should see a blue screen with options in a grey box in the centre, like so:

ADDRESS: 302 PARANJPE UDYOG BHVAN, ABOVE KHANDELWAL SWEETS, THANE STATION ROAD(W)
PHONE : 8097071144/8097071155

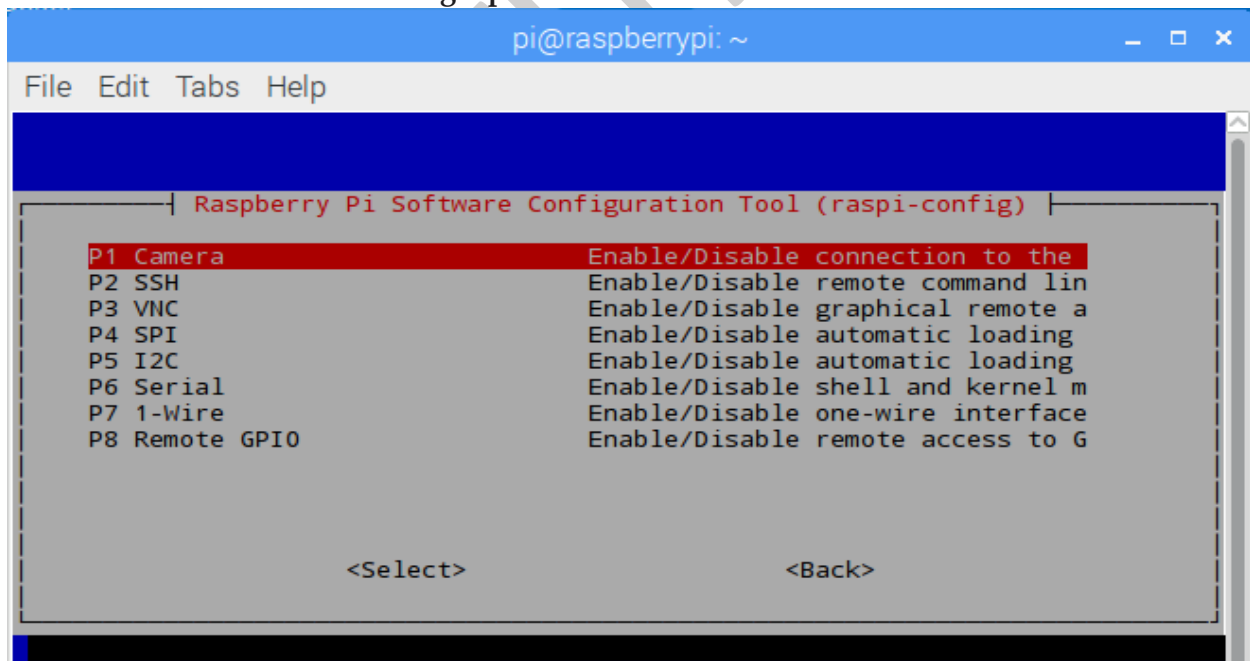


In main menu select 3 Boot Options





In main menu select 5 Interfacing Options



UNIT 2

About Raspbian

Raspbian is a free operating system based on Debian optimized for the Raspberry Pi hardware. An operating system is the set of basic programs and utilities that make your Raspberry Pi run. However, Raspbian provides more than a pure OS: it comes with over 35,000 packages, pre-compiled software bundled in a nice format for easy installation on your Raspberry Pi.

The initial build of over 35,000 Raspbian packages, optimized for best performance on the Raspberry Pi, was completed in June of 2012. However, Raspbian is still under active development with an emphasis on improving the stability and performance of as many Debian packages as possible.

Raspbian is not affiliated with the Raspberry Pi Foundation. Raspbian was created by a small, dedicated team of developers that are fans of the Raspberry Pi hardware, the educational goals of the Raspberry Pi Foundation and, of course, the Debian Project.

Raspbian was created by Mike Thompson and Peter Green as an independent project.

The operating system is still under active development. Raspbian is highly optimized for the Raspberry Pi line's low-performance ARM CPUs.

Raspbian uses PIXEL, Pi Improved Xwindows Environment, Lightweight as its main desktop environment as of the latest update. It is composed of a modified LXDE desktop environment and the Openbox stacking window manager with a new theme and few other changes.

Linux Commands

FILESYSTEM

LS

The ls command lists the content of the current directory (or one that is specified). It can be used with the -l flag to display additional information (permissions, owner, group, size, date and timestamp of last edit) about each file and directory in a list format. The -a flag allows you to view files beginning with . (i.e. dotfiles).

CD

Using `cd` changes the current directory to the one specified. You can use relative (i.e. `cd directoryA`) or absolute (i.e. `cd /home/pi/directoryA`) paths.

PWD

The `pwd` command displays the name of the present working directory: on a Raspberry Pi, entering `pwd` will output something like `/home/pi`.

MKDIR

You can use `mkdir` to create a new directory, e.g. `mkdir newDir` would create the directory `newDir` in the present working directory.

RMDIR

To remove empty directories, use `rmdir`. So, for example, `rmdir oldDir` will remove the directory `oldDir` only if it is empty.

RM

The command `rm` removes the specified file (or recursively from a directory when used with `-r`). Be careful with this command: files deleted in this way are mostly gone for good!

CP

Using `cp` makes a copy of a file and places it at the specified location (this is similar to copying and pasting). For example, `cp ~/fileA /home/otherUser/` would copy the file `fileA` from your home directory to that of the user `otherUser` (assuming you have permission to copy it there). This command can either take `FILE FILE` (`cp fileA fileB`), `FILE DIR` (`cp fileA /directoryB/`) or `-r DIR DIR` (which recursively copies the contents of directories) as arguments.

MV

The `mv` command moves a file and places it at the specified location (so where `cp` performs a 'copy-paste', `mv` performs a 'cut-paste'). The usage is similar to `cp`. So `mv ~/fileA /home/otherUser/` would move the file `fileA` from your home directory to that of the user `otherUser`. This command can either take `FILE FILE` (`mv fileA fileB`), `FILE DIR` (`mv fileA /directoryB/`) or `DIR DIR` (`mv /directoryB /directoryC`) as arguments. This command is also useful as a method to rename files and directories after they've been created.

TOUCH

ADDRESS: 302 PARANJPE UDYOG BHVAN, ABOVE KHANDELWAL SWEETS, THANE STATION ROAD(W)
PHONE : 8097071144/8097071155

The command touch sets the last modified time-stamp of the specified file(s) or creates it if it does not already exist.

CAT

You can use cat to list the contents of file(s), e.g. cat thisFile will display the contents of thisFile. Can be used to list the contents of multiple files, i.e. cat *.txt will list the contents of all .txt files in the current directory.

HEAD

The head command displays the beginning of a file. Can be used with -n to specify the number of lines to show (by default ten), or with -c to specify the number of bytes.

TAIL

The opposite of head, tail displays the end of a file. The starting point in the file can be specified either through -b for 512 byte blocks, -c for bytes, or -n for number of lines.

CHMOD

You would normally use chmod to change the permissions for a file. The chmod command can use symbols u (user that owns the file), g (the files group) , and o (other users) and the permissions r (read), w (write), and x (execute). Using chmod u+x *filename* will add execute permission for the owner of the file.

CHOWN

The chown command changes the user and/or group that owns a file. It normally needs to be run as root using sudo e.g. sudo chown pi:root *filename* will change the owner to pi and the group to root.

SSH

ssh denotes the secure shell. Connect to another computer using an encrypted network connection. For more details see SSH (secure shell)

SCP

The scp command copies a file from one computer to another using ssh. For more details see SCP (secure copy)

SUDO

The sudo command enables you to run a command as a superuser, or another user. Use sudo -s for a superuser shell. For more details see Root user / sudo

DD

The dd command copies a file converting the file as specified. It is often used to copy an entire disk to a single file or back again. So, for example, dd if=/dev/sdd of=backup.img will create a backup image from an SD card or USB disk drive at /dev/sdd. Make sure to use the correct drive when copying an image to the SD card as it can overwrite the entire disk.

DF

Use df to display the disk space available and used on the mounted filesystems. Use df -h to see the output in a human-readable format using M for MBs rather than showing number of bytes.

UNZIP

The unzip command extracts the files from a compressed zip file.

TAR

Use tar to store or extract files from a tape archive file. It can also reduce the space required by compressing the file similar to a zip file.

To create a compressed file, use tar -cvzf *filename.tar.gz* *directory/* To extract the contents of a file, use tar -xvzf *filename.tar.gz*

PIPES

A pipe allows the output from one command to be used as the input for another command. The pipe symbol is a vertical line |. For example, to only show the first ten entries of the ls command it can be piped through the head command ls | head

TREE

Use the tree command to show a directory and all subdirectories and files indented as a tree structure.

&

Run a command in the background with &, freeing up the shell for future commands.

ADDRESS: 302 PARANJPE UDYOG BHVAN, ABOVE KHANDELWAL SWEETS, THANE STATION ROAD(W)

PHONE : 8097071144/8097071155

WGET

Download a file from the web directly to the computer with wget. So `wget https://www.raspberrypi.org/documentation/linux/usage/commands.md` will download this file to your computer as `commands.md`

CURL

Use curl to download or upload a file to/from a server. By default, it will output the file contents of the file to the screen.

MAN

Show the manual page for a file with man. To find out more, run `man man` to view the manual page of the man command.

SEARCH

GREP

Use grep to search inside files for certain search patterns. For example, `grep "search" *.txt` will look in all the files in the current directory ending with `.txt` for the string search.

The grep command supports regular expressions which allows special letter combinations to be included in the search.

AWK

awk is a programming language useful for searching and manipulating text files.

FIND

The find command searches a directory and subdirectories for files matching certain patterns.

WHEREIS

Use whereis to find the location of a command. It looks through standard program locations until it finds the requested command.

NETWORKING

PING

ADDRESS: 302 PARANJPE UDYOG BHVAN, ABOVE KHANDELWAL SWEETS, THANE STATION ROAD(W)
PHONE : 8097071144/8097071155

The ping utility is usually used to check if communication can be made with another host. It can be used with default settings by just specifying a hostname (e.g. ping raspberrypi.org) or an IP address (e.g. ping 8.8.8.8). It can specify the number of packets to send with the -c flag.

NMAP

nmap is a network exploration and scanning tool. It can return port and OS information about a host or a range of hosts. Running just nmap will display the options available as well as example usage.

HOSTNAME

The hostname command displays the current hostname of the system. A privileged (super) user can set the hostname to a new one by supplying it as an argument (e.g. hostname new-host).

IFCONFIG

Use ifconfig to display the network configuration details for the interfaces on the current system when run without any arguments (i.e. ifconfig). By supplying the command with the name of an interface (e.g. eth0 or lo) you can then alter the configuration: check the manual page for more details.

Configuring Raspberry Pi with Linux Commands

INSTALLING SOFTWARE THROUGH APT-GET

Rather than using the Pi Store to download new software you can use the command apt-get, this is the 'package manager' that is included with any Debian based Linux distributions (including Raspbian).

It allows you to install and manage new software packages on your Pi.

In order to install a new package you would type **sudo apt-get install <package-name>** (where <package-name> is the package that you want to install).

Running **sudo apt-get update** updates a list of software packages that are available on your system.

If a new version of a package is available then **sudo apt-get upgrade** will update any old packages to the new version. Finally,

sudo apt-get remove <package-name> removes or uninstalls a package from your system.

Programing interfaces

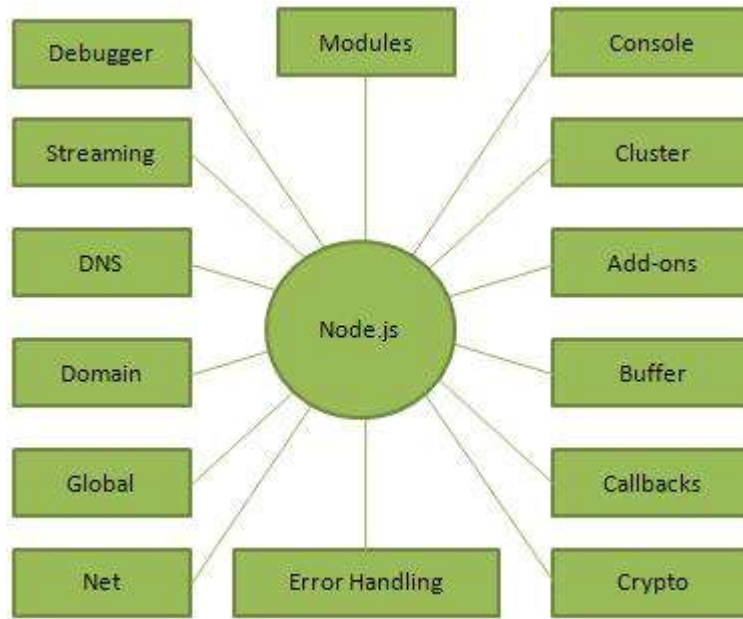
Introduction to Node.js

- Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine) for easily building fast and scalable network applications
- Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.
- Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications.
- Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.
- Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.
- Node.js = Runtime Environment + JavaScript Library

Features of Node.js

- **Asynchronous and Event Driven** – All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
- **Very Fast** – Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- **Single Threaded but Highly Scalable** – Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
- **No Buffering** – Node.js applications never buffer any data. These applications simply output the data in chunks.
- **License** – Node.js is released under the MIT license (Massachusetts Institute of Technology)

The following diagram depicts some important parts of Node.js



Where to Use Node.js?

- I/O bound Applications
- Data Streaming Applications
- Data Intensive Real-time Applications (DIRT)
- JSON APIs based Applications
- Single Page Applications

How to install node.js

```
$ sudo apt install nodejs
```

We can then test and see what version of Node we are running and launch the Node

```
$ node -v
v8.1.2
$ node
> 1 + 3
4
```

Python

The Python IDLE is basically a text editor that lets you execute Python code. If you want to use Python as a server-side language, you certainly can. Python can output HTML just like other languages can, but Python is more commonly used as a module rather than intertwined like some PHP or ColdFusion.

Its comes pre-installed in Raspbian OS

Raspberry Pi Interfaces

UART - Universal Asynchronous Receiver/Transmitter

UART is an asynchronous serial communication protocol, meaning that it takes bytes of data and transmits the individual bits in a sequential fashion.

Asynchronous transmission allows data to be transmitted without the sender having to send a clock signal to the receiver. Instead, the sender and receiver agree on timing parameters in advance and special bits called 'start bits' are added to each word and used to synchronize the sending and receiving units.

UART is commonly used on the Pi as a convenient way to control it over the GPIO, or access the kernel boot messages from the serial console (enabled by default).

It can also be used as a way to interface an Arduino, bootloaded ATmega, ESP8266, etc with your Pi.

Enable UART

A new property has been introduced to enable the UART on the Pi. This property will put the core frequency to a minimum, ensuring stability. It's possible to put the core frequency to maximum as well, assuming the power supply is powerful enough and the Pi 3 is properly cooled (heatsink!).

Enabling UART with minimum core frequency:

```
pi@raspberrypi:~ $ sudo nano /boot/config.txt
# Enable UART
enable_uart=1
```

Enabling UART with maximum core frequency:

```
pi@raspberrypi:~ $ sudo nano /boot/config.txt
```

```
# Enable UART
enable_uart=1
force_turbo=1
```

AN INTRODUCTION TO GPIO AND PHYSICAL COMPUTING ON THE RASPBERRY PI

One powerful feature of the Raspberry Pi is the row of GPIO (general purpose input/output) pins along the top edge of the board.

These pins are a physical interface between the Pi and the outside world. At the simplest level, you can think of them as switches that you can turn on or off (input) or that the Pi can turn on or off (output). Of the 40 pins, 26 are GPIO pins and the others are power or ground pins (plus two ID EEPROM pins which you should not play with unless you know your stuff!)

WHAT ARE THEY FOR? WHAT CAN I DO WITH THEM?

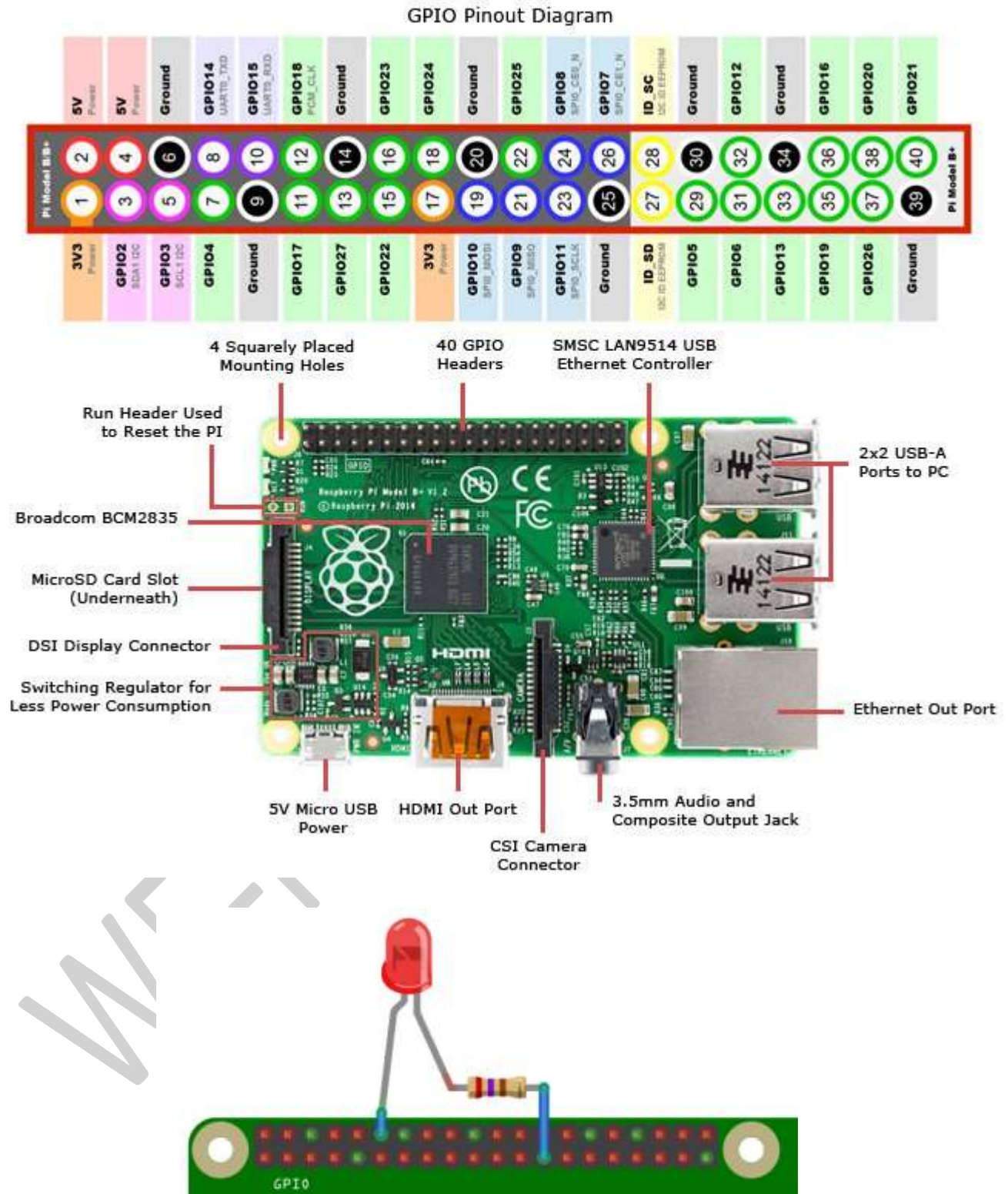
You can program the pins to interact in amazing ways with the real world. Inputs don't have to come from a physical switch; it could be input from a sensor or a signal from another computer or device, for example. The output can also do anything, from turning on an LED to sending a signal or data to another device. If the Raspberry Pi is on a network, you can control devices that are attached to it from anywhere** and those devices can send data back. Connectivity and control of physical devices over the internet is a powerful and exciting thing, and the Raspberry Pi is ideal for this

HOW THE GPIO PINS WORK

OUTPUT

When we use a GPIO pin as an output, the Raspberry Pi replaces both the switch and the battery in the above diagram. Each pin can turn on or off, or go HIGH or LOW in computing terms. When the pin is HIGH it outputs 3.3 volts (3v3); when the pin is LOW it is off.

Here's the same circuit using the Raspberry Pi. The LED is connected to a GPIO pin (which can output +3v3) and a ground pin (which is 0v and acts like the negative terminal of the battery)



I2C and SPI

There are many peripherals that can be added to a microprocessor over the I2C and SPI serial interfaces

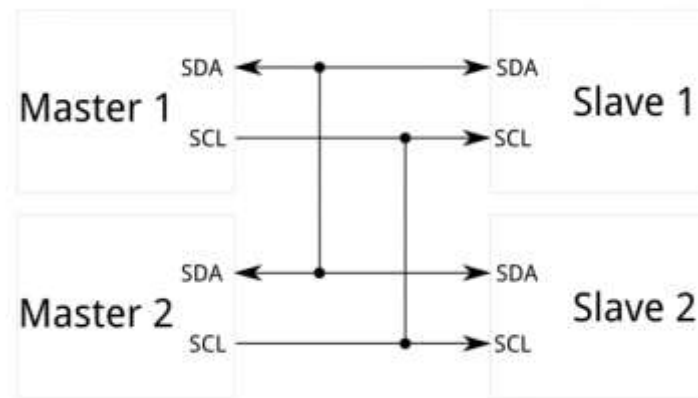
I2C

The Inter-integrated Circuit (I2C) Protocol is a protocol intended to allow multiple “slave” digital integrated circuits (“chips”) to communicate with one or more “master” chips.

I2C requires a mere two wires, like asynchronous serial, but those two wires can support up to 1008 slave devices. Also, unlike SPI, I2C can support a multi-master system, allowing more than one master to communicate with all devices on the bus (although the master devices can’t talk to each other over the bus and must take turns using the bus lines).

Data rates fall between asynchronous serial and SPI; most I2C devices can communicate at 100kHz or 400kHz. There is some overhead with I2C; for every 8 bits of data to be sent, one extra bit of meta data (the “ACK/NACK” bit, which we’ll discuss later) must be transmitted.

The hardware required to implement I2C is more complex than SPI, but less than asynchronous serial. It can be fairly trivially implemented in software.



SPI

Serial Peripheral Interface (SPI) is an interface bus commonly used to send data between microcontrollers and small peripherals such as shift registers, sensors, and SD cards. It uses separate clock and data lines, along with a select line to choose the device you wish to talk to.

Problems with serial port

A common serial port, the kind with TX and RX lines, is called “asynchronous” (not synchronous) because there is no control over when data is sent or any guarantee that both sides are running at precisely the same rate. Since computers normally rely on everything being synchronized to a single “clock” (the main crystal attached to a computer that drives everything), this can be a problem when two systems with slightly different clocks try to communicate with each other.

To work around this problem, asynchronous serial connections add extra start and stop bits to each byte help the receiver sync up to data as it arrives. Both sides must also agree on the transmission speed (such as 9600 bits per second) in advance. Slight differences in the transmission rate aren’t a problem because the receiver re-syncs at the start of each byte.

Solution

SPI works in a slightly different manner. It’s a “synchronous” data bus, which means that it uses separate lines for data and a “clock” that keeps both sides in perfect sync. The clock is an oscillating signal that tells the receiver exactly when to sample the bits on the data line. This could be the rising (low to high) or falling (high to low) edge of the clock signal; the datasheet will specify which one to use.

Advantages of SPI:

- It’s faster than asynchronous serial
- The receive hardware can be a simple shift register
- It supports multiple slaves

Disadvantages of SPI:

- It requires more signal lines (wires) than other communications methods
- The communications must be well-defined in advance (you can’t send random amounts of data whenever you want)
- The master must control all communications (slaves can’t talk directly to each other)
- It usually requires separate SS lines to each slave, which can be problematic if numerous slaves are needed.

Useful Implementations

cross compiler

A cross compiler is a compiler that runs on one platform/architecture but generates binaries for another platform/architecture. With devices like the Raspberry Pi, where you really don’t have

much CPU or memory to work with, if you're doing any heavy compiling (like when working on the kernel) a cross compiler is the only way to go.

For example, I build all my Raspberry Pi kernels on my nice Sandy Bridge Xeon E3 home server where they compile in only a fraction of the time they would on the Pi.

While there are a lot of different methods for building cross-compilers, by far the quickest and easiest is to use crosstool-ng. This is a set of scripts that bring up a menuconfig-like interface to choose your compiler settings, then goes off and downloads what it needs, patches it, configures it, builds it and installs it all for you.

Installing crosstool-ng on centos

```
yum -y install bison flex gperf libtool texinfo gcc gcc-c++ gmp-devel ncurses-devel
```

Get the latestest crosstool-ng from <http://crosstool-ng.org/>

```
cd /usr/src
wget http://crosstool-ng.org/download/crosstool-ng/crosstool-ng-x.y.z.tar.bz2
tar jxvf crosstool-ng-1.20.0.tar.bz2
cd crosstool-ng-1.20.0
```

Build the tool

```
mkdir -p /opt/crosstool-ng
chown -r nobody:nobody /opt/crosstool-ng
chmod g+w,g+s /opt/crosstool-ng
./configure --prefix=/opt/crosstool-ng
make && make install
```

Setup autocompletion

```
cp ct-ng.comp /etc/bash_completion.d/
```

Add crosstool's bin directory to the user's path

```
PATH=$PATH:/opt/crosstool-ng-1.20.0/bin
```

Configure the toolchain

Re-login as regular user, or source ~/.bash_profile

Setup the config directory

ADDRESS: 302 PARANJPE UDYOG BHVAN, ABOVE KHANDELWAL SWEETS, THANE STATION ROAD(W)
PHONE : 8097071144/8097071155

```
mkdir -p /opt/croostool-ng/config  
cd /opt/croostool-ng/config
```

Run the configuration tool

```
./ct-ng menuconfig
```

Setup the following Items

- Paths and misc options
 - Enable Try features marked as EXPERIMENTAL"
 - Set the "Prefix directory" from "\${HOME}/x-tools/\${CT_TARGET}" to "/opt/croostool-ng/tools/\${CT_TARGET}", please note that the /opt/croostool-ng/config directory will contain the configuration, downloaded tools, temporary build files and the /opt/croostool-ng/tools will contain your actual toolchain.
- Toolchain options
 - Change tuple's version string from the default "unknown" to "bashlinux"
- Target options
 - Be sure that the "Target Architecture" is set to "arm"
 - Be sure that "Little Endian" and "32bit" are selected
 - Be sure that "Floating point" is set to "hardware (FPU)"
 - Be sure that "Use EABI" is selected
 - Be sure that "Append 'hf' to the tupe" is selected
- Operating system
 - Set "Target OS" to Linux
- C compiler
 - Enable "Show Linaro versions"
 - The "gcc version" should automatically change to latest linaro version, if not then set it manually. Currently the latest is "linaro-4.8-2014.01"
 - Enable "C++" in order to have C++ compiler
 - Disable "Link libstdc++ statically into gcc binary" otherwise you will get the the following error "[ERROR] Static linking impossible on the host system 'x86_64-build_unknown-linux-gnu'"

- Disable "Enable GRAPHITE loop optimisations" otherwise you will get the the following error "Installing PPL for host [ERROR] configure: error: Cannot find GMP version 4.1.3 or higher."

Build the tool

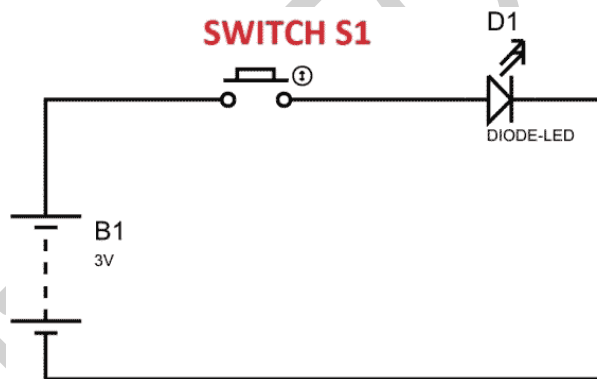
```
ct-ng build
```

Append the PATH with the new tools

```
PATH=$PATH:/opt/crosstool-ng/tools/arm-bashlinux-linux-gnueabi/bin
```

ref: <http://wiki.bashlinux.com/Crosstool-ng>

Pulse Width Modulation



In above figure, if the switch is closed continuously over a period of time, the LED will be 'ON' during this time continuously. If the switch is closed for half second and opened for next half second, then LED will be ON only in the first half second. Now the proportion for which the LED is ON over the total time is called the Duty Cycle, and can be calculated as follows:

Duty Cycle = Turn ON time / (Turn ON time + Turn OFF time)

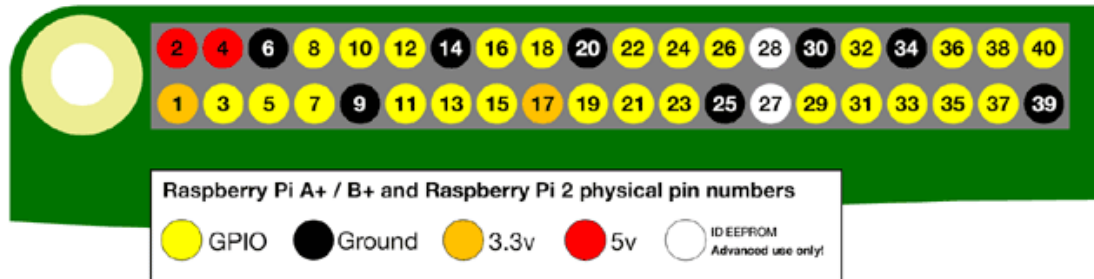
Duty Cycle = (0.5 / (0.5 + 0.5)) = 50%

So the average output voltage will be 50% of the battery voltage.

This is the case for one second and we can see the LED being OFF for half second and LED being ON the other half second. If Frequency of ON and OFF times increased from '1 per second' to '50 per second'. The human eye cannot capture this frequency. For a normal eye the

LED will be seen, as glowing with half of the brightness. So with further reduction of ON time the LED appears much lighter.

We will program the PI for getting a PWM and connect a LED to show its working.



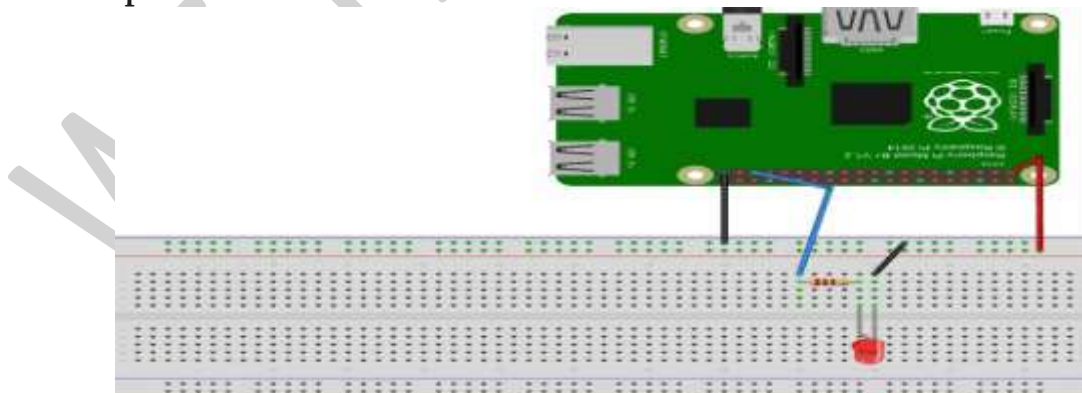
There are 40 GPIO output pins in Raspberry Pi. But out of 40, only 26 GPIO pins (GPIO2 to GPIO27) can be programmed.

Components Required:

Here we are using Raspberry Pi 2 Model B with Raspbian Jessie OS. All the basic Hardware and Software requirements are previously discussed, you can look it up in the Raspberry Pi Introduction, other than that we need

- Connecting pins
- 220Ω or 1KΩ resistor
- LED
- Bread Board

Circuit Explanation:



Working Explanation:

Once everything is connected, we can turn ON the Raspberry Pi to write the program in PYTHON and execute it.

We will talk about few commands which we are going to use in PYTHON program.

We are going to import GPIO file from library, below function enables us to program GPIO pins of PI. We are also renaming "GPIO" to "IO", so in the program whenever we want to refer to GPIO pins we will use the word 'IO'.

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

Sometimes, when the GPIO pins, which we are trying to use, might be doing some other functions. In that case, we will receive warnings while executing the program. Below command tells the PI to ignore the warnings and proceed with the program.

```
IO.setwarnings(False)
```

We can refer the GPIO pins of PI, either by pin number on board or by their function number. In pin diagram, you can see 'PIN 35' on the board is 'GPIO19'. So we tell here either we are going to represent the pin here by '35' or '19'.

```
IO.setmode (IO.BCM)
```

We are setting GPIO19 (or PIN35) as output pin. We will get PWM output from this pin.

```
IO.setup(19, IO.OUT)
```

After setting the pin as output we need to setup the pin as PWM output pin,

```
p = IO.PWM(output channel , frequency of PWM signal)
```

The above command is for setting up the channel and also for setting up the frequency of the PWM signal. 'p' here is a variable it can be anything. We are using GPIO19 as the PWM *output channel*. '*frequency of PWM signal*' has been chosen 100, as we don't want to see LED blinking.

Below command is used to start PWM signal generation, '*DUTYCYCLE*' is for setting the Turn On ratio, 0 means LED will be ON for 0% of time, 30 means LED will be ON for 30% of the time and 100 means completely ON.

```
p.start(DUTYCYCLE)
```

This command executes the loop 50 times, x being incremented from 0 to 49.

for x in range (50):

While 1: is used for infinity loop. With this command the statements inside this loop will be executed continuously.

With the program being executed, the duty cycle of PWM signal increases. And then decreases after reaching 100%. With an LED attached to this PIN, brightness of LED increases first and then decreases.

Code:

```
import RPi.GPIO as IO      #calling header file which helps us use GPIO's of PI
import time                #calling time to provide delays in program
IO.setwarnings(False)     #do not show any warnings
IO.setmode (IO.BCM)       #we are programming the GPIO by BCM pin numbers. (PIN35 as
'GPIO19')
IO.setup(19,IO.OUT)        # initialize GPIO19 as an output.
p = IO.PWM(19,100)         #GPIO19 as PWM output, with 100Hz frequency
p.start(0)                 #generate PWM signal with 0% duty cycle

while 1:                   #execute loop forever

    for x in range (50):   #execute loop for 50 times, x being incremented from 0 to 49.
        p.ChangeDutyCycle(x) #change duty cycle for varying the brightness of LED.
        time.sleep(0.1)      #sleep for 100m second

    for x in range (50):   #execute loop for 50 times, x being incremented from 0 to 49.
        p.ChangeDutyCycle(50-x) #change duty cycle for changing the brightness of LED.
        time.sleep(0.1)      #sleep for 100m second
```


UNIT 3

introduction to IoT

What is IoT

The Internet of things (IoT) is the inter-networking of physical devices, vehicles (also referred to as "connected devices" and "smart devices"), buildings, and other items embedded with electronics, software, sensors, actuators, and network connectivity which enable these objects to collect and exchange data

In 2013 the Global Standards Initiative on Internet of Things (IoT-GSI) defined the IoT as "a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies and for these purposes a "thing" is "an object of the physical world (physical things) or the information world (virtual things), which is capable of being identified and integrated into communication networks".

The IoT allows objects to be sensed or controlled remotely across existing network infrastructure, creating opportunities for more direct integration of the physical world into computer-based systems, and resulting in improved efficiency, accuracy and economic benefit in addition to reduced human intervention.

When IoT is augmented with sensors and actuators, the technology becomes an instance of the more general class of cyber-physical systems, which also encompasses technologies such as smart grids, virtual power plants, smart homes, intelligent transportation and smart cities.

IoT examples

Smart thermostats

Imagine that you can control the temperature of your home from anywhere, with just a simple touch on your smart phone or tablet, and you get the desired temperature before getting home so that you don't need to wait

Mimo Monitor:

They have been around for some time now. Mimo Monitors are not only affordable but also presents a new technology. Used for several business purposes, this IoT has been making things easier, simpler and is said to be very productive. But now Mimo Monitors offers something unique and unexpected. The technology now enables you get updated to your baby's body position, their breathing level, body temperature, response to activities and health. The information collected is intimated to you through your mobile phones. Further the above explained, "Nest Learning Thermostat", can automatically change the temperature of your baby's nursery, according to the data, got by Mimo meter.

Philips-Hue Bulbs

Philips hue bulbs have now stepped into a new stage of innovation with these smart bulbs. Linked with your mobile phones, you can now actually control the intensity of lights on your fingertips. The combination of bulb with mobile technology is next thing for your home. Instead of going for different watt of bulbs to suit the mood and the environment, simply change the intensity from dim to medium to full using your phone. These bulbs can be programmed to get dim at night, also it can work as a alarm by setting it in blinking mode on any intruder detection. The lighting can be dynamically change according to the environment, like a different lighting when watching a movie. ON and OFF timer can also be set for these bulbs to automatically ON and OFF after a particular time.

Ralph Lauren Polotech Shirt

Ralph Lauren is a American clothing company which has launched The Polotech Shirt for athletes and become a pioneer to bring IoT in clothing industry. This Shirt can record the biometric readings of Athletes like Heart rate, calories burned, activity levels, breathing depths etc. and can help him to deliver the best performance. It can be connected to the Apple watch or an iPhone, and can track & record all the activities in your iPhone. So this Polotech Shirt along with the iPhone can become your complete fitness tracker or we can say fitness trainer.

led example using python

Required List of components

- a. 1 - Raspberry Pi device with 5 V power supply.
- b. 1 - Bread Board
- c. 1 - Male to Female jumper cable
- d. 1 - LED

Pin#	NAME	Connection	Connection	NAME	Pin#
01	3.3V		5V (Cupcade)	5V	02
03	GPIO 2		5V (Powerboost)	5V	04
05	GPIO 3		GND (Powerboost)	Ground	06
07	GPIO 4	START		GPIO 14	08
09	Ground	GND (Cupcade)		GPIO 16	10
11	GPIO 17	UP	SELECT	GPIO 18	12
13	GPIO 27	DOWN	GND (Select/Start)	Ground	14
15	GPIO 22	LEFT	RIGHT	GPIO 23	16
17	3.3V		A	GPIO 24	18
19	GPIO 10	B	GND (ABXYR)	Ground	20
21	GPIO 09	X	Y	GPIO 25	22
23	GPIO 11	L Shoulder	R Shoulder	GPIO 08	24
25	Ground	GND (L)		GPIO 07	26
27	ID_SD			ID_SC	28
29	GPIO 05			Ground	30
31	GPIO 06			GPIO 12	32
33	GPIO 13			Ground	34
35	GPIO 19			GPIO 16	36
37	GPIO 26			GPIO 20	38
39	Ground			GPIO 21	40

connect led pins according to polarity (as per above pins)

python code

```

1. import RPi.GPIO as GPIO
2. import time
3. #assign numbering for the GPIO using BCM
4. GPIO.setmode(GPIO.BCM)
5. #assignn number for the GPIO using Board
6. #GPIO.setmode(GPIO.BOARD)
7.
8. cnt = 0
9. MAIL_CHECK_FREQ = 1 # change LED status every 1 seconds
10. RED_LED = 4
11. GPIO.setup(RED_LED, GPIO.OUT)
12. while True:
13.     if cnt == 0 :
14.         GPIO.output(RED_LED, False)
15.         cnt = 1
16.     else:
17.         GPIO.output(RED_LED, True)
18.         cnt = 0
19.
20.     time.sleep(MAIL_CHECK_FREQ)

```

21. `GPIO.cleanup()`

IoT and Protocols

The HTTP Protocol

It is safe to assume that most people that use a computer today have had an experience of **Hypertext Transfer Protocol (HTTP)**, perhaps without even knowing it. When they "surf the Web", what they do is they navigate between pages using a browser that communicates with the server using HTTP.

HTTP has become much more than navigation between pages on the Internet.

Today, it is also used in **machine to machine (M2M)** communication, automation, and Internet of Things, among other things.

So much is done on the Internet today, using the HTTP protocol, because it is easily accessible and easy to relate to. For this reason, we are starting our study of Internet of Things by studying HTTP.

This will allow you to get a good grasp of its strengths and weaknesses, even though it is perhaps one of the more technically complex protocols.

HTTP basics

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 basically 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. This can be observed in the following diagram:



HTTP request/response pattern

Each resource, originally thought to be a collection of Hypertext documents or HTML documents, is identified by a **Uniform Resource Locator (URL)**. Clients simply use the `GET` method to request a resource from the corresponding server. In the structure of the URL presented next, the resource is identified by the path and the server by the authority portions of the URL.

HTTP defines a set of headers that can be used to attach meta information about the requests and responses sent over the network.

HTTP works on top of the **Internet Protocol (IP)**. In this protocol, machines are addressed using an IP address, which makes it possible to communicate between different **local area networks (LANs)** that might use different addressing schemes, even though the most common ones are Ethernet-type networks that use **media access control (MAC)** addresses.

Communication in HTTP is then done over a **Transmission Control Protocol (TCP)** connection between the client and the server. The TCP connection makes sure that the packets are not lost and are received in the same order in which they were sent.

Encryption can be done through the use of **Secure Sockets Layer (SSL)** or **Transport Layer Security (TLS)**. When this is done, the protocol is normally named **Hypertext Transfer Protocol Secure (HTTPS)** and the communication is performed on a separate port, normally 443. In this case, most commonly the server, but also the client, can be authenticated using X.509 certificates that are based on a **Public Key Infrastructure (PKI)**, where anybody with access to the public part of the certificate can encrypt data meant for the holder of the private part of the certificate.

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.

Introducing UPnP

UPnP is a very common protocol. It is used by almost all network-enabled consumer electronics products used in your home or office, and as such, it is a vital part of **Digital Living Network Alliance (DLNA)**.

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)**.

However, they can also search for the network using multicast addressing for certain types of devices or services. Actions on services are called using SOAP web service calls.

Providing a service architecture

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.

Documenting device and service capabilities

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.

The CoAP Protocol

The UPnP Protocol, we also saw the benefits of simplifying HTTP and using it over UDP instead of TCP. But for tiny resource-constrained devices that communicate over resource-constrained IP networks, HTTPU is not a practical option because it requires too many resources and too much bandwidth.

This is especially the case when the underlying network limits the size of datagrams, which is the case when you use IPv6 over Low power Wireless Personal Area Networks (6LoWPAN), a radio networking protocol based on the latest version of Internet Protocol Version 6 (IPv6).

Constrained Application Protocol (CoAP) is an attempt to solve this.

Making HTTP binary

The main difference between CoAP and HTTPU is that CoAP replaces the text headers used in HTTPU with more compact binary headers, and furthermore, it reduces the number of options available in the header.

This makes it much easier to encode and parse CoAP messages. CoAP also reduces the set of methods that can be used; it allows you to have four methods: GET, POST, PUT, and DELETE.

Also, in CoAP, method calls can be made using confirmable and nonconfirmable 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 each format is identified by a number instead of its corresponding Internet Media Type.

A detailed list of the numbers assigned to different options, methods, status code, and Content-Formats used in CoAP

CoAP Method Codes

Code	Name
0.01	GET
0.02	POST
0.03	PUT
0.04	DELETE
0.05	FETCH
0.06	PATCH
0.07	iPATCH
0.08-0.31	Unassigned

CoAP Response Codes

Code	Description
2.00	Unassigned
2.01	Created
2.02	Deleted
2.03	Valid
2.04	Changed
2.05	Content
2.06-2.30	Unassigned
2.31	Continue
3.00-3.31	Reserved
4.00	Bad Request
4.01	Unauthorized
4.02	Bad Option

Like with HTTPU, CoAP supports multicasting.

This can be used to detect devices or communicate through firewalls

CoAP 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 through the use of Datagram Transport Layer Security (DTLS)

The MQTT Protocol

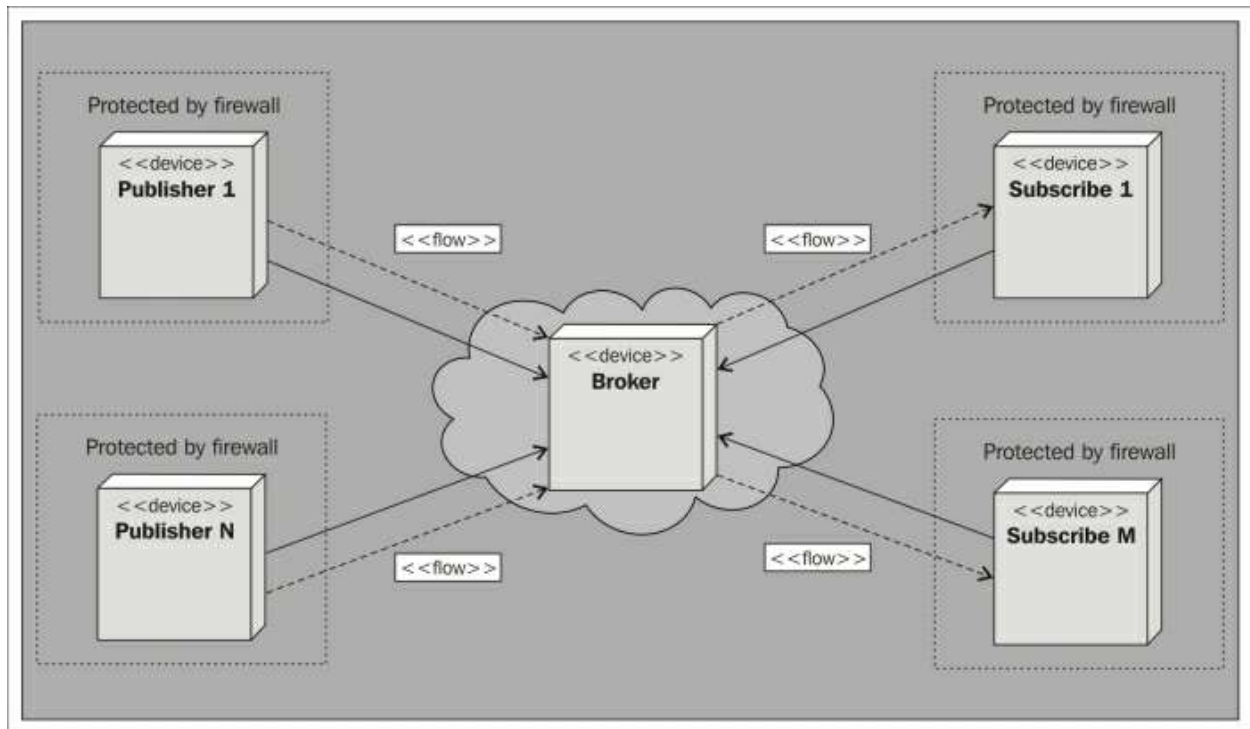
One of the major problems we encountered when we looked at the HTTP, UPnP, and CoAP protocols is how to cross firewall boundaries

Firewalls not only block incoming connection attempts, but they also hide a home or office network behind a single IP address.

Unless the firewall blocks outgoing connections, which it does only if explicitly configured to do so, we can cross firewall boundaries if all the endpoints in a conversation act as clients to a common message broker that lies outside of the firewall and is therefore accessible to everybody.

The message broker acts as a server, but all it does is relay messages between clients

One protocol that uses message brokers is the Message Queue Telemetry Transport (MQTT) protocol.



Publishing and subscribing

The MQTT protocol is based on the publish/subscribe pattern

The publish/subscribe pattern has three types of actors:

- **Publisher:** The role of the publisher is to connect to the message broker and publish content
- **Subscriber:** They connect to the same message broker and subscribe to content that they are interested in
- **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 content should be retained by the server or not.

If retained, each subscriber will receive the latest published value directly when subscribing.

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's also a single-level wildcard character: the plus character (+).

As an example, to illustrate topic semantics, our sensor will publish measured temperature on theClayster/LearningIoT/Sensor/Temperature topic.

Subscribing to Clayster+/Sensor/# will subscribe to all the subbranches of the Sensor class that start with Clayster/, then any subtopic, which in turn will have a Sensor/ subtopic.

There are three Quality of Service levels in MQTT available while publishing content.

1. The lowest level is an unacknowledged service. Here, the message is delivered at most once to each subscriber.
2. 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.
3. 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.

The support for user authentication in MQTT is weak.

Plain text username and password authentication exists, but it provides an obvious risk if the server is not hosted in a controlled environment

To circumvent the most obvious problems, MQTT can be used over an encrypted connection using SSL/TLS.

In this case, it is important for clients to validate server certificates else user credentials may be compromised.

The XMPP Protocol

MQTT protocol is limited to a single communication pattern: the publish/subscribe pattern.

This is useful in cases where a thing only publishes data and has many consumers of its data, and where data is homogenous and most reported data items are actually used.

If individually tailored messages, momentary values, or real-time or bidirectional communication is important, or if data is seldom used compared to the frequency with which it is updated, other communication patterns would be more appropriate.

Extensible Messaging and Presence Protocol (XMPP) protocol. The XMPP protocol also uses message brokers to bypass firewall barriers. But 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 experience.

XMPP basics

XMPP was originally designed for use in instant messaging applications (or chat). It is an open protocol, standardized by **Internet Engineering Task Force (IETF)**, as are HTTP and CoAP. Even though it was first designed for chat applications, it lends itself very well to other applications, such as the ones for IoT, due to its flexibility and richness of communication patterns.

The XMPP architecture builds on the tremendous success and global scalability of the **Simple Mail Transfer Protocol (SMTP)**

The difference is that XMPP is designed for real-time instantaneous messaging applications, where smaller messages are sent with as little latency as possible and without any persistence

XMPP uses a federated network of XMPP servers as message brokers to allow clients behind separate firewalls to communicate with each other

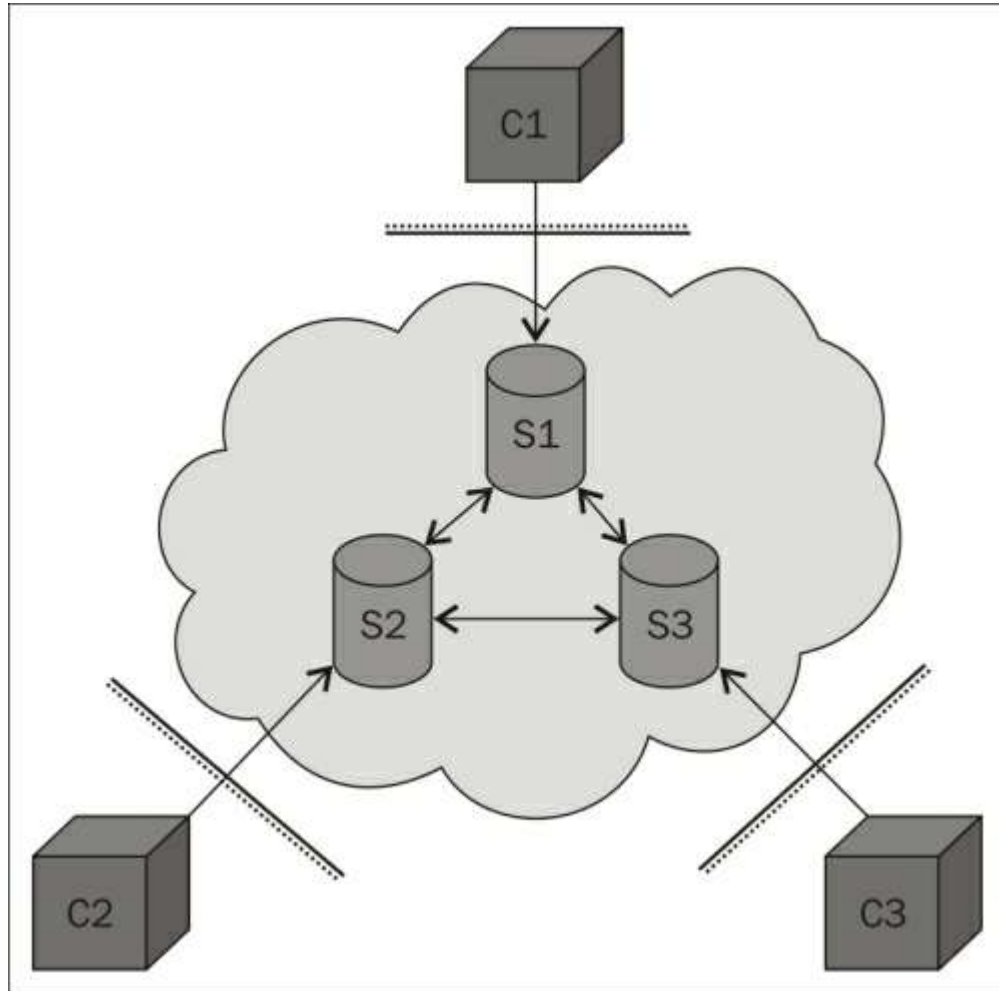
Each server controls its own domain and authenticates users on that domain.

Clients can communicate with clients on other domains through the use of federation where the servers create connections between themselves in a secure manner to interchange messages between their domains.

It is this federation that allows you to have a globally scalable architecture. All of this happens at the server level, so there is nothing that clients need to worry about. 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.

It is this architecture of federation that makes XMPP scalable and allows you to make billions of devices communicate with each other in the same federated network.

The following illustration shows how clients (**C1**, **C2**, and **C3**) behind firewalls connect to different servers (**S1**, **S2**, and **S3**) in a federated network to exchange messages:



A small federated XMPP network

Providing a global identity

XMPP servers do more than relay messages between clients. They also provide each client with an authenticated identity.

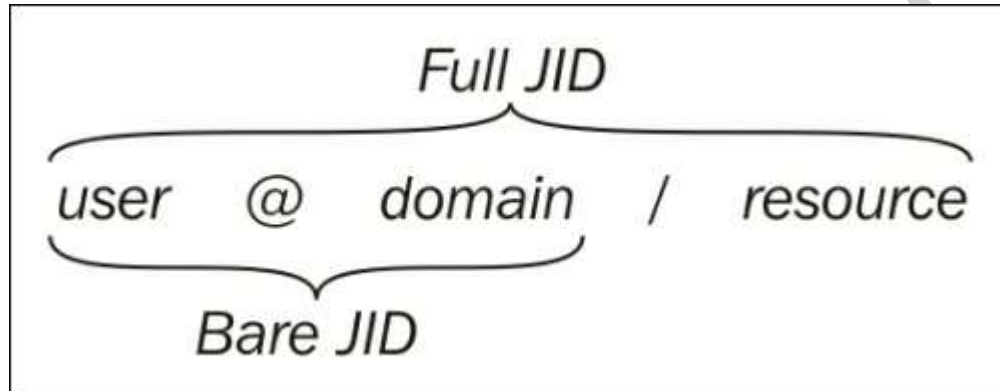
If the server is a public server in the global federated network of XMPP servers, it is a global identity.

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, encrypting the communication between them. The identity of the client is often called XMPP address or **Jabber ID (JID)**.

Each connection is also bound to a specific resource, which is normally a random string. Together, the username, domain name, and resource constitute the full JID of a connection, while only the username and domain name constitute the bare JID of an account.



Authorizing communication

Another important reason for using XMPP servers to relay communication instead of serverless peer-to-peer technologies is to assure the clients that only authorized communication will be relayed. This feature comes in handy, especially 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 or device behind it. The reason is twofold:

- First, multiple clients might use the same account at the same time. You need to 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.

Sensing online presence

To learn about the resource part of a corresponding client, you send a presence subscription to its bare JID. If accepted by the remote client, you will receive presence messages every time the state of the contact is changed, informing you whether it is online, offline, away, busy, and so on.

In this presence message, you will also receive the full JID of the contact. Once a presence subscription has been accepted by the remote device, it might send you a presence subscription of its own, which you can either accept or reject. If both the parties accept it and subscribe to the presence from each other, then parties are said to be friends.

Communication patterns

XMPP supports a rich set of communication patterns. It does this by providing three communication primitives called stanzas

- The First is the presence stanza. This is used to send information about oneself to interested and authorized parties.
- The second is the message stanza. This is used to send asynchronous messages to a given receiver.
- The third is the iq stanza, short for information/query. This stanza is used to provide a request/response communication pattern.

Extending XMPP

To improve interoperability, the source code for the implementation of these extensions has also been included in the **Clayster.Library.IoT** library. These extensions include extensions to communicate with sensor data or control actuators, sending asynchronous events based on subscriber-specific conditions. They also include extensions to register devices securely and provision the services in the networks.

Instead, data will be made available on request and only given to a trusted few by the owner-approved parties.

An alternative way to connect to an XMPP server is by using Bidirectional streams Over Synchronous HTTP (BOSH). This allows clients with access to only the HTTP protocol to use XMPP as well. Some servers also publish XMPP over web socket interfaces. This makes it possible to access the XMPP network for clients, such as web browsers and so on.

XMPP servers also receive connections from other XMPP servers. This is part of the federation feature of XMPP. These servers connect to each other using the xmpp-server service

Provisioning for added security

creation of identity, where things by themselves create their own identity on the network. Once a thing has created an identity on the network, we will introduce a way to register the thing, discover it, and safely claim ownership of it. Once the ownership has been claimed, we can then use the provisioning extension to delegate trust to a trusted third party, a provisioning server, which we will then use to control who can connect to our devices and what they can do

IoT Service as a Platform

Clayster platform

We will start the download of the Clayster platform by downloading its version meant for private use from <http://www.clayster.com/downloads>.

Console Applications outlines the process to create a simple console application. When we create a service for a service platform, the executable EXE file already exists.

Therefore, we have to create a library project instead and make sure that the target framework corresponds to the version of the Clayster distribution (dot net 3.5)

Such a project will generate a **dynamic link library (DLL)** file. During startup, the Clayster executable file will load any DLL file it finds marked as a *Clayster Module* in its installation folder or any of its subfolders.

libraries available in the Clayster distribution

- **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.

ADDRESS: 302 PARANJPE UDYOG BHVAN, ABOVE KHANDELWAL SWEETS, THANE STATION ROAD(W)
PHONE : 8097071144/8097071155

- **Clayster.Library.Meters:** It contains an abstraction model for things such as sensors, actuators, controllers, meters, and so on

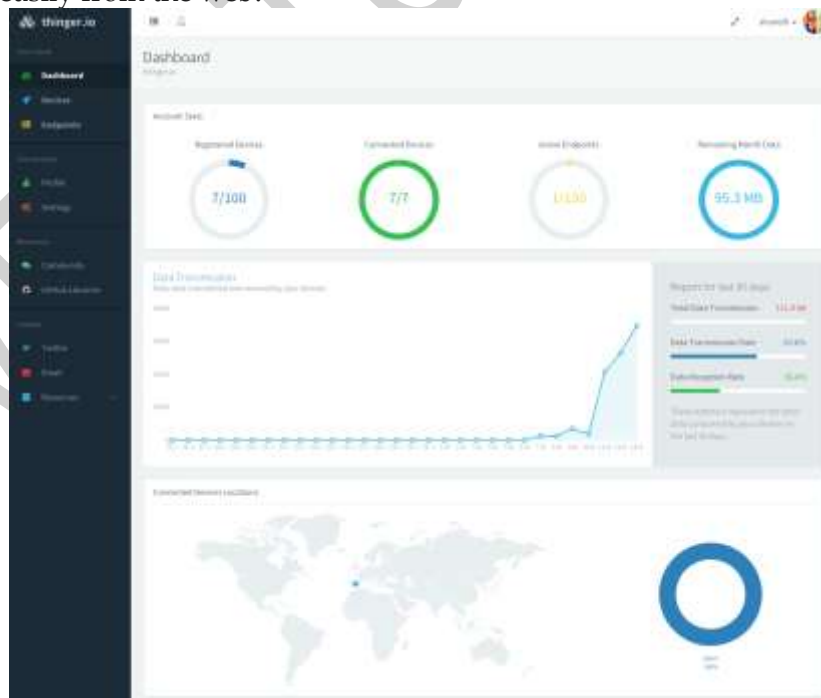
Apart from the libraries, we will also add two additional references to the project—this time to two service modules available in the distribution, which are as follows:

- **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.Meters** namespace.

thinger.io

Description

- Thinger.io is a platform that allows connecting things to the Internet. And what's new? Hardware agnostic: Connect anything! from basic Arduinos with very limited resources, to more complex embedded systems running linux like Raspberry Pi
- It is Open Source, so you can take the code and build your own cloud if you want. It provides thing API discovery right out of the box, so you can code your things and interact easily from the web.



thinger.io platform in the Raspberry Pi

Installing a newer GCC Version

```
sudo apt-get install gcc-4.9 g++-4.9
```

Install additional dependencies

```
sudo apt-get install cmake libssl-dev (Install Dependencies (CMake and OpenSSL))
```

Starting with the platform

Download the latest Linux Client version from GitHub.

```
git clone https://github.com/thinger-io/Linux-Client.git
```

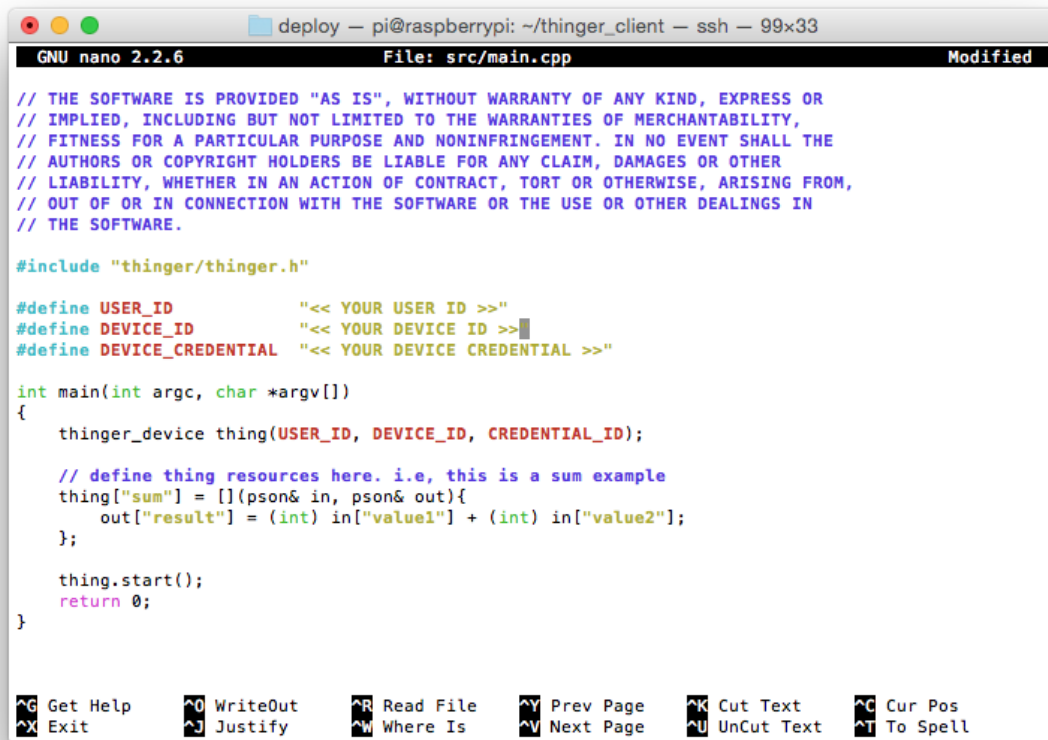
Enter in the Linux-Client folder we just cloned.

```
cd Linux-Client
```

And now it is time to enter the credentials in the main project file. So we are going to edit the main.cpp file from src folder. You can use any editor you want. We will use here nano.

```
nano src/main.cpp
```

In this case you must edit the fields USER_ID, DEVICE_ID, and DEVICE_CREDENTIAL with the information you provided while registering your device in the platform. Here is an example screenshot of how the main.cpp file should look like before editing these fields.



```
GNU nano 2.2.6 File: src/main.cpp Modified

// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
// THE SOFTWARE.

#include "thinger/thinger.h"

#define USER_ID " << YOUR USER ID >>"
#define DEVICE_ID " << YOUR DEVICE ID >>"
#define DEVICE_CREDENTIAL " << YOUR DEVICE CREDENTIAL >>"

int main(int argc, char *argv[])
{
    thinger_device thing(USER_ID, DEVICE_ID, CREDENTIAL_ID);

    // define thing resources here. i.e, this is a sum example
    thing["sum"] = [] (pson& in, pson& out){
        out["result"] = (int) in["value1"] + (int) in["value2"];
    };

    thing.start();
    return 0;
}

^G Get Help      ^O WriteOut      ^R Read File     ^Y Prev Page     ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify       ^W Where Is     ^V Next Page     ^U UnCut Text    ^T To Spell
```

When you are done editing the parameters, exit the nano editor pressing `Ctrl+X` and then type 'y' to save the changes.

Now you must add execute permissions to the `run.sh` script.

```
chmod +x run.sh
```

And now you can run it to test that everything is working.

```
./run.sh
```

If everything is going fine you should see how the program is compiled and executed automatically.

```
deploy — pi@raspberrypi: ~/thinger_client — ssh — 99x33
CMakeLists.txt  README.md  install  run.sh  src
pi@raspberrypi ~/thinger_client $ nano src/main.cpp
pi@raspberrypi ~/thinger_client $ chmod +x run.sh
pi@raspberrypi ~/thinger_client $ ./run.sh
-- The C compiler identification is GNU 4.9.2
-- The CXX compiler identification is GNU 4.9.2
-- Check for working C compiler: /usr/bin/gcc
-- Check for working C compiler: /usr/bin/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Performing Test COMPILER_SUPPORTS_CXX11
-- Performing Test COMPILER_SUPPORTS_CXX11 - Success
-- Performing Test COMPILER_SUPPORTS_CXX0X
-- Performing Test COMPILER_SUPPORTS_CXX0X - Success
-- Found OpenSSL: /usr/lib/arm-linux-gnueabi/libssl.so;/usr/lib/arm-linux-gnueabi/libcrypto.so
(found version "1.0.1e")
-- Configuring done
-- Generating done
-- Build files have been written to: /home/pi/thinger_client/build
Scanning dependencies of target thinger
[100%] Building CXX object CMakeFiles/thinger.dir/src/main.cpp.o
Linking CXX executable thinger
[100%] Built target thinger
[2562342.778000]: Not connected!
[2562342.830000]: Connecting to iot.thinger.io:25202 ...
[2562343.573000]: Connected!
[2562343.574000]: Authenticating...
[2562343.734000]: Authenticated!
```

For test the device resources, please go to the API Explorer that appears in the device dashboard.



SenseIoT

SenseIoT is a platform for storing and processing sensor data safely and securely in the cloud. It comes with tools for visualizing data, setting triggers, and remotely managing data. The SenseIoT API enables developers to integrate the platform and its functions with any devices, sensors, or sensor networks that can connect to the internet.

About the API

The SenseIoT API conforms to the [REST](#) design principles and is therefore based on HTTP Requests. The HTTP requests that are supported are: **GET**, **PUT**, **POST**, **DELETE**, and **OPTIONS**. These methods are for listing, updating, creating, and deleting data respectively. The OPTIONS method enables third party web clients to do [cross-domain requests](#).

PUT / DELETE Alternative

Clients that are not able to perform PUT or DELETE requests can use the POST parameter **_METHOD** to specify the request method. An example how to delete a sensor via a POST request:

```
https://api.sense-os.nl/sensors/1?_METHOD=DELETE
```

Getting Started with the API

The SenseIoT API is based on HTTP requests, and so you need to be able to perform your own, customized API requests to interact with the API.

Step 1: Creating an Account

Step 2: Log In

Step 3: Create a Sensor

Step 4: Upload Data

Now that we have a sensors, lets give it some data. We can upload multiple data points at the same time, and give every data point a (UNIX) timestamp. If you don't give your data points a timestamp, SenseIoT will timestamp them at the time they are received.

Step 5: Retrieve Data

Your application will probably want to retrieve some data to for instance plot it on your screen or to notify you when its hot enough outside to go out for ice-cream. When retrieving data, there are a number of different options, which we will not all go into right now, but be sure to check the [API Reference](#)!

Client Libraries

We have created a couple of libraries to help you write client applications that access the SenseIoT API. There are different libraries for different platforms and languages. The libraries are open source, so feel free to incorporate them into your own app.

iOS and Android Clients

The libraries for iOS and Android are a special case: they also give you easy access to the sensors on the phone. They contain everything you need to build a context aware app

Available Client Libraries

- Android: source, tutorial;
- iOS: source, tutorial;
- Python: source, tutorial.
- Ruby: source, tutorial.

Carriots

Carriots is an application hosting and development platform ([Platform as a Service](#)) specially designed for projects related to the [Internet of Things](#) (IoT) and [Machine to Machine](#) (M2M).

Enables data collection from connected objects (the things part), store it, build powerful applications with few lines of code and integration with external IT systems (the internet part).

Carriots provides a development environment, APIs and hosting for IoT projects development.

Functionalities

Device management: Remotely maintain, control and interact with devices regardless of their location. Check status, change configurations, enable/disable or upgrade devices' firmware.

Listeners: When data is received, when it is stored, when a device is connected or disconnected...

Rules: Complex scripts, reusable pieces of code or simple ready to use logic.

Triggers: Pushes data to an external system with Carriots' triggers. A method to implement data brokerage.

SDK application engine: Listeners and rules, both are executed by Carriots SDK engine. Sandboxing Java execution threads to keep secure and efficient. All the DB is accessible from the scripts.

Data export: Integrate with other IT systems, export files creation, data push to other databases or REST API usage to manage outbound data.

Custom alarms: Alarms are created by Carriots when something goes wrong. Custom Alarms creation and management are used to notify, acknowledge and discard for custom notifications flow.

User management: Corporate users can manage other user accounts to manage projects or to define custom defined privileges and visibility.

Custom control panel: Custom control panel creation with any technology to manage all Carriots entities using REST API.



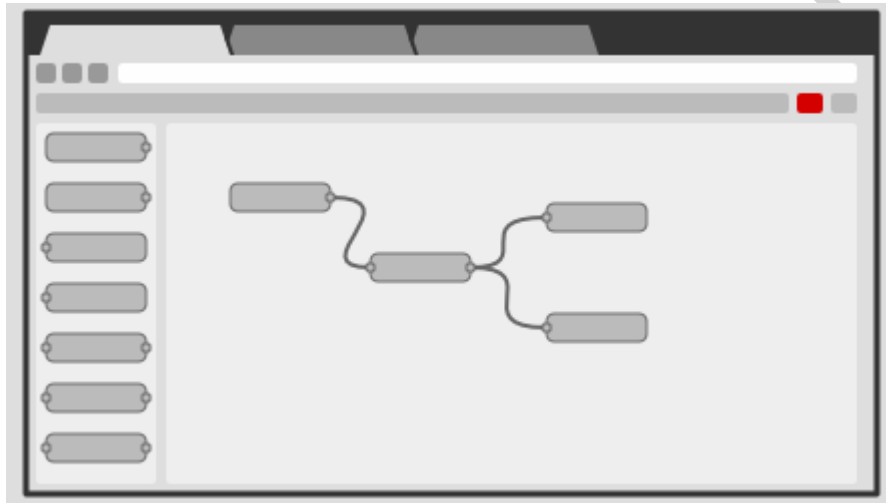
Node-RED

Node-RED is a software tool developed by [IBM](#) for wiring together hardware devices, [APIs](#) and [online services](#) as part of the [Internet of Things](#).

Node-RED provides a browser-based flow editor, which can be used to create [JavaScript](#) functions. Elements of applications can be saved or shared for re-use. The runtime is built on [Node.js](#). The flows created in Node-RED are stored using [JSON](#).

Node-RED is built on Node.js, taking full advantage of its event-driven, non-blocking model. This makes it ideal to run at the edge of the network on low-cost hardware such as the Raspberry Pi as well as in the cloud.

By default, the Node-RED editor is not secured - anyone who can access the IP address and port it is running on can access the editor and deploy changes. This is only suitable if you are running on a trusted network.



Security and Interoperability

Understanding the risks

There are many solutions and products marketed today under the label IoT that lack basic security architectures.

It is very easy for a knowledgeable person to take control of devices for malicious purposes.

It is just a matter of time before somebody is literally killed as a result of an attack by a hacker on some vulnerable equipment connected to the Internet.

While management knows how to manage known risks, they don't know how to measure them in the field of IoT and computer communication. This makes them incapable of understanding the consequences of architectural decisions made by its engineers.

The engineers in turn might not be interested in focusing on risks, but on functionality, which is the fun part.

generation of engineers who tackle **IoT** are not the same type of engineers who tackled application development on the Internet.

engineers actually re-use solutions and previous experience, they don't really fit well in many cases. The old communication patterns designed for web applications on the Internet are not applicable for **IoT**.

Knowing your neighbor

it might be a good idea to know your neighbors first. It's the same when you move a **M2M** application to **IoT**. As soon as you connect the cable, you have billions of neighbors around the world, all with access to your device.

What kind of neighbors are they? Even though there are a lot of nice and ignorant neighbors on the Internet, you also have a lot of criminals, con artists, perverts, hackers.

Modes of attack

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.

Guessing the credentials

One way to get access to a system is to impersonate a client in the system by trying to guess the client's credentials. To make this type of attack less effective, make sure each client and each device has a long and unique, perhaps randomly generated, set of credentials.

Getting access to stored credentials

One common way to illicitly enter a system is when user credentials are found somewhere else and reused.

Often, people reuse credentials in different systems. There are various ways to avoid this risk from happening.

One is to make sure that credentials are not reused in different devices or across different services and applications.

Man in the middle

Another way to gain access to a system is to try and impersonate a server component in a system instead of a client. This is often referred to as a **Man in the middle (MITM)** attack.

The reason for the middle part is that the attacker often does not know how to act in the server and simply forwards the messages between the real client and the server.

To avoid this type of attack, it's important for all clients (not just a few) to always validate the identity of the server it connects to. If it is a high-value entity, it is often identified using a certificate.

Sniffing network communication

If communication is not encrypted, everybody with access to the communication stream can read the messages using simple sniffing applications, such as Wireshark.

Remember to always use encryption if sensitive data is communicated. If data is private, encryption should still be used, even if the data might not be sensitive at first glance.

Port scanning and web crawling

Port scanning is a method where you systematically test a range of ports across a range of IP addresses to see which ports are open and serviced by applications.

This method can be combined with different tests to see the applications that might be behind these ports.

If HTTP servers are found, standard page names and web-crawling techniques can be used to try to figure out which web resources lie behind each HTTP server.

Breaking ciphers

Ciphers can be broken using known vulnerabilities in code where attackers exploit program implementations rather than the underlying algorithm of the cipher.

To safeguard against such attacks, it's important to realize that an attacker does not spend more effort into an attack than what is expected to be gained by the attack.

By storing massive amounts of sensitive data centrally or controlling massive amounts of devices from one point, you increase the value of the target, increasing the interest of attacking it.

Tools for achieving security

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)**. Often, traditional **M2M** solutions working well in local intranets need 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.

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.

X.509 certificates and encryption

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.

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.

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.

Username and passwords

A common method to provide user credentials during authentication is by providing a simple username and password to the server. This is a very human concept. Some solutions use the concept of a pre-shared key (PSK) instead, as it is more applicable to machines, conceptually at least.

device creates its own random identity and creates the corresponding account in the XMPP server in a secure manner.

Referred websites

<http://www.c-sharpcorner.com/UploadFile/015c2d/raspberry-pi-simple-led-blinking-program-using-python/>

<http://thisdavej.com/beginners-guide-to-installing-node-js-on-a-raspberry-pi/>

<http://www.raspberrypi-spy.co.uk/2014/08/enabling-the-spi-interface-on-the-raspberry-pi/>

<https://www.raspberrypi.org/documentation/hardware/raspberrypi/spi/README.md>

https://wiki.archlinux.org/index.php/Raspberry_Pi#SPI

<https://www.raspberrypi.org/forums/viewtopic.php?f=98&t=113425>

<http://www.bootc.net/archives/2012/05/26/how-to-build-a-cross-compiler-for-your-raspberry-pi/>

<https://www.modmypi.com/blog/loading-i2c-spi-and-1-wire-drivers-on-the-raspberry-pi-under-raspbian-wheezy>