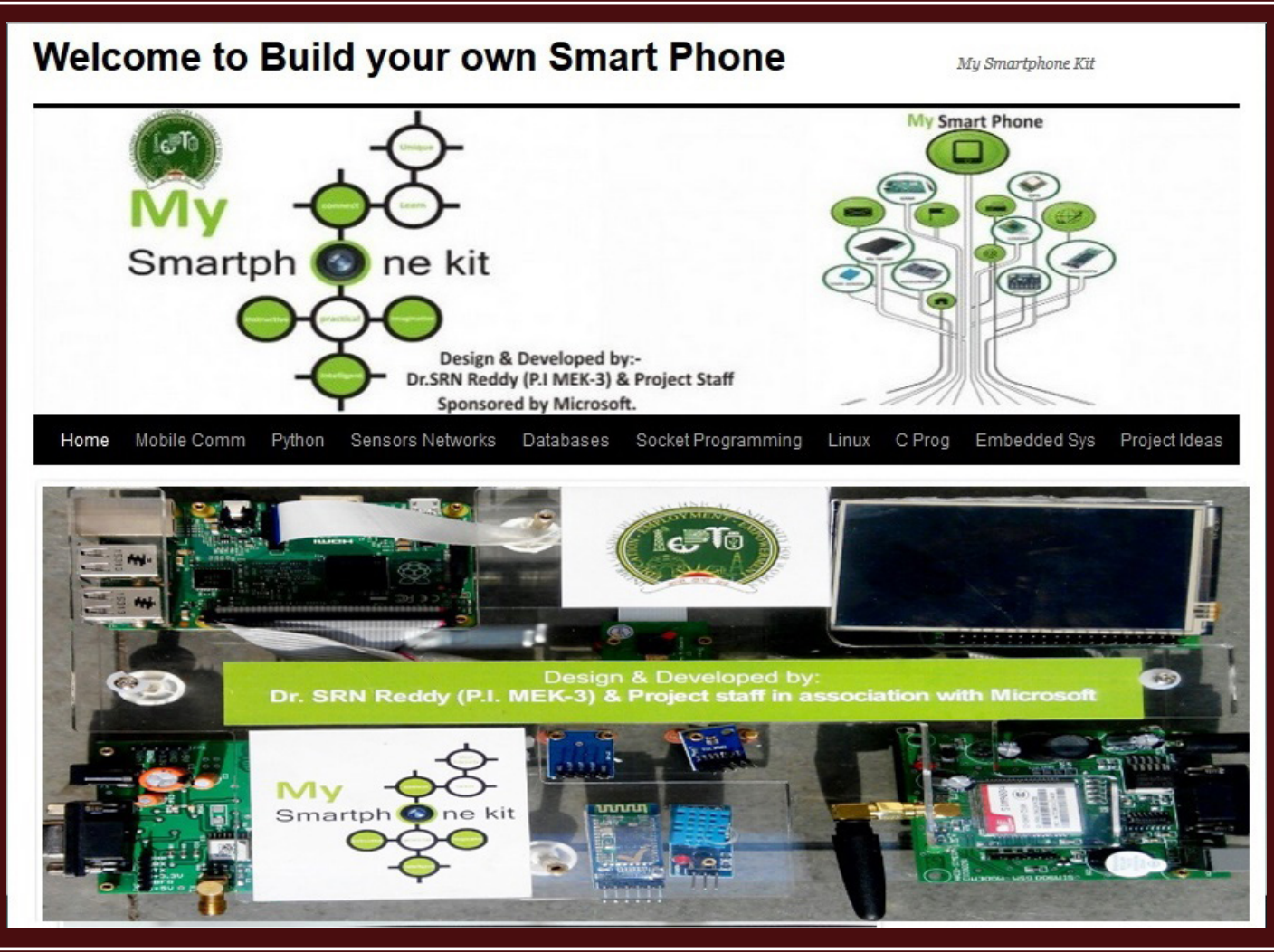
Design and Development of Smart Phone

A Practical Approach by using Raspberry Pi2



Dr. SRN Reddy
Mr. Suresh Chande

Design and Development of Smart Phone

A Practical Approach by using Raspberry Pi2

Design and Development of Smart Phone

A Practical Approach by using Raspberry Pi2

Dr. SRN Reddy

Suresh Chande

SHROFF PUBLISHERS & DISTRIBUTORS PVT. LTD.

Mumbai *Bangalore* *Kolkata* *New Delhi*

Design and Development of Smart Phone: A Practical Approach by using Raspberry Pi2

Dr. SRN Reddy, Suresh Chande

First Edition: January 2016

ISBN : 978-93-5213-281-2

Published by **Shroff Publishers & Distributors Pvt. Ltd.** C-103, TTC Industrial Area, MIDC, Pawane, Navi Mumbai - 400 703 • TEL: (91 22) 4158 4158 • FAX: (91 22) 4158 4141\ E-mail: spdorders@shroffpublishers.com • Web: www.shroffpublishers.com CIN : U22200MH1992PTC067760 Printed at Jasmine Art Printers Pvt. Ltd., Navi Mumbai.

To my students who believed that we can make it..

Foreword

There has been an exponential growth in mobile and hand held devices for the last decade. This has resulted into penetration of mobile devices and its applications in all the domains of all the users ranging from common man to a richest person in the world. Mobile computing hardware and software technologies are not only used in the industry but also in the academics and research. There exist several courses in undergraduate and postgraduate level to teach mobile computing, mobile application development, mobile architecture & programming. However, the available text books only describe the theoretical aspects of mobile communications, mobile computing but does not address the practical aspects.

The book 'Design and Development of Smart Phone - A Practical Approach by using Raspberry Pi' attempts to provide a blend of theoretical and more on practical aspects with real hardware and software as an integrated complete solution. The approach used is unique methodology that helps the students and researchers to understand the current technologies with practical exposure that leads to build a low cost Smartphone affordable to all the students. This further enhances the online web support for both hardware, software and application development with the help of www.mobileeducationkit.net a sponsored projects by Nokia, Microsoft, Intel and MHRD of India together to build a practical exposure in the cutting edge technologies like design and development of mobile devices, build your own smart device etc.

Prof. Nupur Prakash
Vice Chancellor
Indira Gandhi Delhi Technical University for Women
Kashmere Gate, Delhi-06

Preface

The book 'Design and Development of Smart Phone - A Practical Approach by using Raspberry Pi'(mobile education kit) is a unique in its kind since it provides complete package that student or a researcher requires to understand, experiment and create new products and innovations through the theoretical and practical knowledge provided by the book. There exist several books that talks about mainly on theoretical concepts or simulations but fails to address a real practical approach with the real hardware, software. This book describes the step by step procedure to integrate different hardware and software components to finally develop a unique product of low cost smart phone. This book uses one of the most popular open source language 'python', a open source OS platform 'Linux' and open source hardware Raspberry Pi which are accepted across the world. This book deals with understanding the architecture of mobile device in terms of both hardware and software. It describes the selection of various SoCs, operating systems and programming languages in order to develop the smart phone. It also describes the methodology for porting of customized Linux operating system as a mobile OS to support various functionalities with the raspberry pi. The subsequent chapters describes the interfacing and programming of various I/O and communication peripherals such as TFT, Mike, Speakers ,various sensors and GSM Module, GPS Module, Bluetooth modules respectively along with examples and exercises.

This book supports online web portal "www. Mobileeducationkit.net" with recent updates on software, experiments that provides a round the clock supportive environment for beginners to learn and experiment. The aim of the book is to enhance the practical approach with the real environment among the students and researchers in the open source environment. The mobile education kit is distributed along with the software packages that comprise the OS source code, project files that demonstrate the various experiments and ppts for the faculties with the list of exercises and solutions. The hardware comprises the raspberry pi, micro SD card, communication module like GSM, GPS, Bluetooth with various sensors. Updates on any of the topic are made available through the web portal www.mobileeducationkit.net. Amy student/researcher or faculty can contribute for the next version of the release by their name and it will be made available by their name if found suitable.

Acknowledgements

I wish to take this opportunity to express my sincere gratitude to the Honourable Vice Chancellor, Indira Gandhi Delhi Technical University for Women, Prof. Nupur Prakash, for her encouragement, constant support during the project and providing me the administrative and research facilities.

My sincere thanks to the Prof. M. Balakrishnan, CSE Deptt, IITD, whose support and guidance at every stage of the research work helped me to work on practical approach. I owe many thanks to Prof. B.V.R. Reddy, Dean, University School of Engineering and Technology, IP University for his moral support and guidance throughout my carrier.

I would like to express my sincere thanks to Nokia University Relations-Finland and Microsoft for funding the three research projects worth 60 lacks on mobile education. It would have been impossible to successfully complete this project without their support. I would like to express my sincere thanks to MHRD & ITRA Govt of India for the sponsored research projects which also helped me to complete the book. Last but not lease I want to express my sincere thanks to my M.Tech , PhD students and research staff who have helped me in creating the several experiments and testing the same in real hardware and made available on www.mobileeducationkit.net. I wish to extend my special thanks to my family, friends and relatives for their moral support.

Design and Development of Smart Phone – A Practical Approach by using Raspberry Pi2

INDEX

Appendix

Chapter 1

Introduction to Mobile Architecture

1.1 Introduction

Rapid proliferation of mobile smartphone technologies has enabled Mobile industry to reach a major milestone. In April 2015, the number of Mobile devices crossed over 7.2 Billion and arguably more than the human population, which is about ~7.19 Billion. The proliferation of mobile phones and particularly emergence of the Smartphones across the globe is significantly enhancing every single person's life in one way or another.

Mobile phones serve a primary purpose of enabling people to stay connected with family and people who matter the most. The global mission of taking voice over wireless in 1990 moved to staying connected to internet in 2000. In 2010 smartphones fueled the growth of rich social networking via mobile applications eco systems backed by Viber, Whatsapp, Line, Facebook to name a few. In the next decade by 2020 Smartphones are considered to emerge into one the major devices covering over 2/3rd of all devices connected to network according to GSMA intelligence.

Mobile phones are positively impacting every part of people's life helping them to stay connected to people, bridge the gap between distance and reducing the need to travel while managing their business more effective. This has led to contributing towards individual's business, nation's economy and in turn growth of global economy. Mobile Phones are packed with multiple advanced technologies to sense and capture our surroundings. This is immensely helping us to be more effectively in life and stay productive in our day to day activities. Mobile users can instantly capture the moment they are experience by means of the camera embedded in the device, listen and view events live or recorded via the internet multimedia service, see TV on the go and manage ones financial and health by means of various services extended via the Mobile devices. Individuals can manage their personal lives more effectively and business while maintaining mobility.

As the smart phone devices are becoming more affordable with the costs ranging from 25$ - 100$ they enable every person in the society to leverage the values offered by mobility. Every person in society, from a street hawker, school children, office workers to the chief of government organizations like Presidents of Nations are benefiting by the proliferation of smartphones. Smartphones can bring a major shift in educating every single person of the society without discrimination as every person connected to the internet can have free access to information they need.

1.1.1Convergence of computing platforms

In early 1990's Mobile industry demonstrated digital cellular technology also known as second generation (2G) mobile hand held device. This was a GSM standard which enabled take mobile call across the globe in a standardized way. The next decade (2000) saw the development of the next generation (3G) technology which enabled mobile industry convergence with the internet world now with a mission of accessing information wirelessly from anywhere. The later evolution of mobile devices created a huge demand for mobile phones to also create information turning a need for phones to get smarter than voice and provide rich capabilities to create and consume rich content leading to the popular era of smartphones.

Smartphones have today turned into a platform with a magnetic effect and have integrated or mobilized multiple adjacent industries such as: Photography with the help of Camera, Multimedia Industry taking Music and Videos. The Broadcasting industry such as Radio and Television also got into the smartphone devices. Navigation systems got integrated into Smartphone devices (GPS). Smartphones are also powered by multiple wireless technologies GSM, Wireless LAN (Wifi), Bluetooth and has housed various Sensors such as touch sensors, light sensors, motion sensors and has the capability to integrate with any sensors seen necessary. Smartphones are actually powerful computers capable to inter-work with existing platforms such as PC, Televisions or Large Screens through software and wireless technologies to utilize external display to extend its own capabilities. Like smartphone users can control a PC remotely or even play Multimedia, stream internet content or recorded content from Smartphone onto larger displays of desktop computers.

Smartphones have arguably been one of the platform which rapidly converged technologies across adjacent industries while adding huge value of mobility and making them ubiquitously accessible across the globe. In addition to direct value to Smartphone users all these technologies packaged onto one product make smartphones one of the best and an excellent platform to study and learn about these technologies. It provides a profound platform for researchers, be it a student, academician, industry or government to create applications, services or even innovate on this platform while delivering value back to the Smartphone users in a rapid cycle. This becomes especially even more appealing as Smartphones are ubiquitous.

1.1.2 Future Trends

Smartphones in the year 2015 are about 70-80% market share in developed markets. While the growth is slowing down in developed markets, it is rapidly increasing in developing markets primarily triggered by the cost of Smartphones dropping down to below $100. This will rapidly lead to smartphones to become ubiquitously available across the world. According to GSMA 2/3rd of all connections will be originated by Smartphones (excluding M2M) globally.
This also means Smartphones will empower many other low end devices and gadgets to connect to the internet. Smartphones will create an ecosystem around itself such as enable watches, near eye displays (Glasses), Wrist Bands packed with sensors to monitor your health, Rings, Your cars' dashboard to utilize the capabilities of the Smartphone to bridge and communicate with the internet world. Smartphones will utilize the capabilities of short range wireless technologies to connect with devices to extend its reach and engage with end users with an entirely amazing new user interactions which is only limited to human imagination. We are already seeing a number of these developments as early demonstration and products being launched.
Mobile computing is at its infancies compared to where we are today. As smartphones are powerful mobile computers they are today utilized to consume and generate content for business and consumer applications and services. For example end users share videos, photos and tweet over social networking site. Users chat and do video calls. Business users join remote meetings and make presentations and work using email client almost all actions they do on a regular PC but limited screen size. As powerful computers they can also be utilized to manage and control home, office and industrial equipment. The smartphones as platform will start to see its capabilities be utilized in entirely new products and solutions we have not seen today.
As smartphone grow to be the largest number of devices connecting to the network they will also

be the largest source of data be it locally stored in the device or distributed to the network. This will pose a big challenge to mobile application and service provider to be ready to scale if a large number of mobile users would utilize the service. For example Cloud computing capabilities are already employed to deliver mobile online services and power various social networking platforms. As mobile devices are today major source of consumer generate data the various development in Data Science and Big Data technologies will need to be employed to make sense of massive amounts of data in order to serve mobile consumers effective.
Smartphones will play an important role in the development of the Internet of Things (IoT). In one way it will enable people with a capability to interact with IoT and in turn IoT will be able to access and ask for human attention via the Smartphones. Internet of Things (IoT) is a concept that every object is provided a uniquely addressable and identifiable element in a network. This object be it physical object, animal, or a person can then transfer data over a network without requiring human-to-human or human-to-computer interaction. It is forecasted that the number of devices connected to Internet of things will rapidly grow from 2.5 Billion device in 2014 to about 24 Billion devices [BI Intelligence] by 2019

Mobile Devices are like Lab in a Bag , as stated by Dr Narayanan, Founder Director of Srishti Institute of Art Design & Technology. People can carry with them anywhere and carry out learning where ever they are in day to day activities. Smartphones houses technologies from many adjacent industries and can serve as a platform for practical learning and teaching. This is true also for anyone who is interested to study photography, telecommunications, Software programming, electronics, computer science and sensor technologies.

This book is a result of several years of efforts to develop a Mobile Education Kit for students to learn about technology and theory in a practical and hands on manner. You will understand in the various chapters of this book how various components work and put them together to see a Smartphone light up at the end of the book. You should not certainly stop there. It really is the beginning of your creation. You should continue to explore various parts of the chapters and go deeper to make a complete deep dive to define your own Smartphone or new form of smart device. It is possible to develop and design your own device to solve a real world problem. This is result of dedicated work over few years of teaching about how to approach and learn about Mobile computing platform with a practical approach. This book should help you understand the details of a Smartphone. As you practice and get your hands on with this book through its online site , we also give you an opportunity as a student or researcher to learn and contribute findings back to the others through the online site www.mobileeducationkit.org. We welcome your contributions be it code snippet, experience sharing or further extensions you make. You can contact the team with your contributions via the book's website.
This chapter helps to understand the concepts, key design principles and attributes of mobile hardware and software architecture. Advances in Embedded Systems, VLSI technology and innovations in hardware and software development have enabled the use of real-time data such as audio, video, animation, music etc. The growing popularity of portable computing, communication and sensing devices with these data types are converging into a single unit named Smartphone or Mobile phone. The future of smart phones lies in convergence of different technologies that supports variety of functions, ease of use, and quality of service with affordable price.

There exist multiple vendors who are specialised in design and development of components of the smart phone components such as operating systems (OS), System on Chip (SoC), communication and computing peripherals. This book helps the students, developers and teachers in understanding the basic components of the Smartphone architecture, interfacing, programming and customizing into a Smartphone in a real environment.

1.2 Mobile Architecture

The basic and detail architecture of the Mobile device is shown in fig 1.1 and fig 1.2 respectively. It consists of Hardware and Software architecture. The main hardware components of the mobile phone is the application processor that controls all other components of the device such as display, keypad, power, audio , video etc. The radio signals are handled by baseband processor which in turn communicates with other processors to use their functionality. Power and audio processor controls the functioning of speaker and microphone with the help of application processor. Subscriber Identification Module (SIM) contains the details about the subscriber.

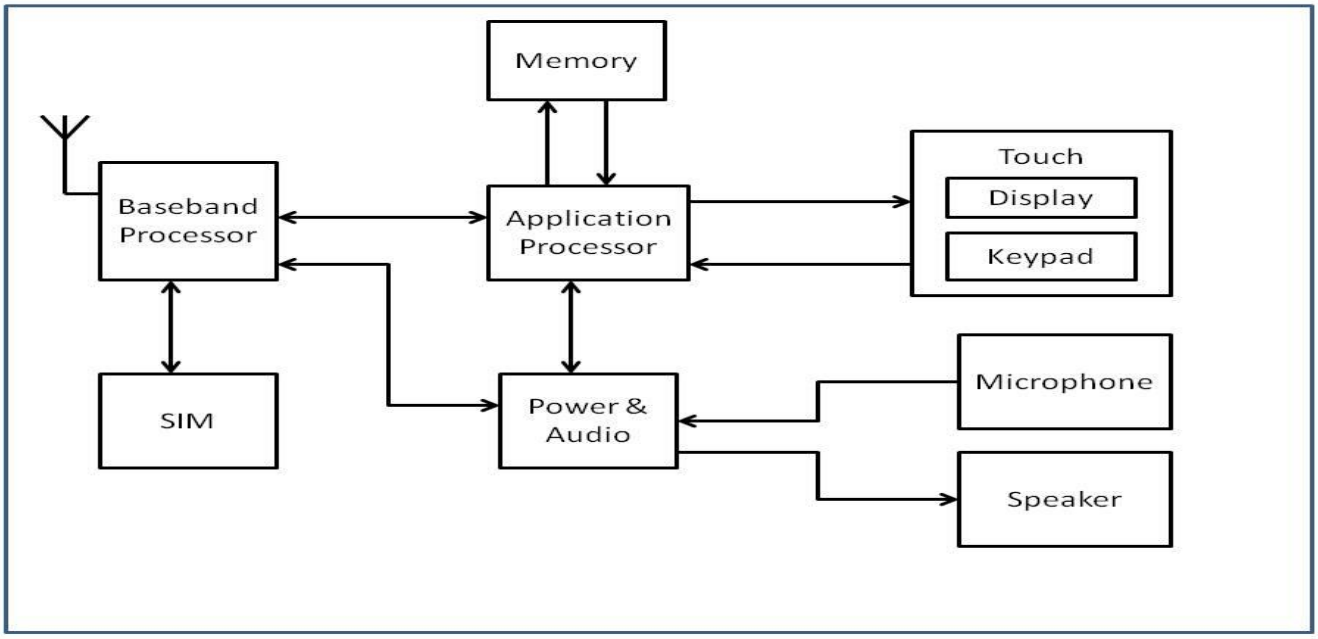


Fig1.1: Basic Block Diagram of Mobile Architecture

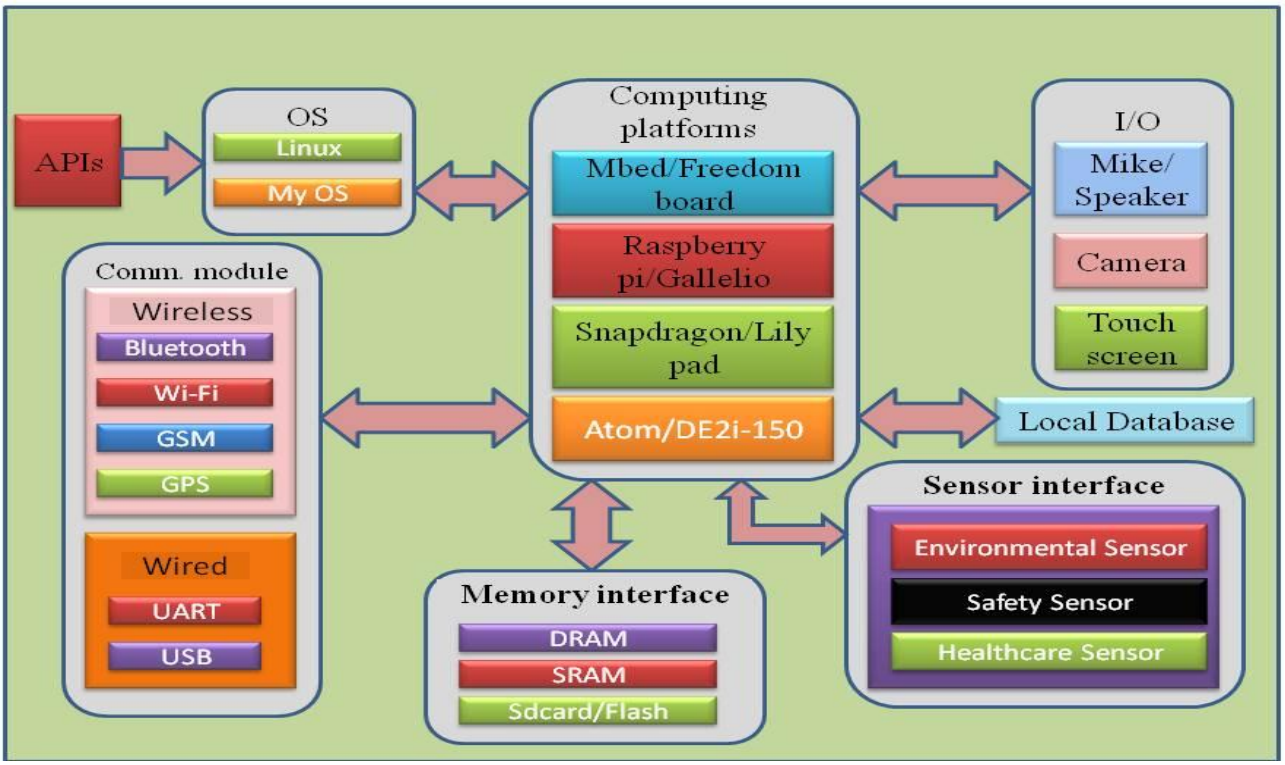


Fig 1.2 Detailed Block Diagram Smartphone

Camera Module A camera is an optical instrument that records images that can be stored directly, transmitted to another location, or both. These images may be still photographs or moving images such as videos or movies. The modern camera evolved from the camera obscure. The functioning of the camera is very similar to the functioning of the human eye.

Memory Card A memory card or flash card is an electronic flash_memory data_storage device used for storing digital information. These are commonly used in many electronic devices, including digital cameras, mobile phones, laptop computers, MP3players and video game consoles, tablets. Most of these can be diminutive, re-recordable, and can retain data without power.

Touch Screen A touch screen is an electronic_visual display that the user can control through simple or multi touch by touching the screen with a special stylus or pen or fingers. The touch screen enables the user to control the applications through user interface (UI).

Sensors A sensor is a device that detects events or changes in quantities and provides a corresponding output, generally as an electrical or optical signal; for example, a thermocouple converts temperature to an output voltage.

1.3 Software Architecture of Smartphone

Framework A software framework is an abstraction in which software providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific software. A software framework is a universal, reusable software environment that provides particular functionality as part of a larger software platform to facilitate development of software applications, products and solutions. Software frameworks may include support programs, compilers, code libraries, tool sets, and application programming interfaces (APIs) that bring together all the different components to enable development of a project or solution.

Kernel The kernel is a computer program that manages input/output requests from software, and translates them into data processing instructions for the central processing unit and other electronic components of a computer. The kernel is a fundamental part of a modern computer's operating system. Because of its critical nature, the kernel code is usually loaded into a protected area of memory, which prevents it from being overwritten by other, less frequently used parts of the operating system or by application programs. The kernel performs its tasks, such as executing processes and handling interrupts, in kernel space, whereas everything a user normally does, such as writing text in a text editor or running programs in a GUI (graphical user interface), is done in user space. This separation is made in order to prevent user data and kernel data from interfering with each other and thereby diminishing performance or causing the system to become unstable (and possibly crashing)

Device Driver A device driver is a program that controls a particular type of device that is attached to your computer. There are device drivers for printers, displays, CD-ROM readers, diskette drives, and so on. When you buy an operating system, many device drivers are built into the product. However, if you later buy a new type of device that the operating system didn't anticipate, you'll have to install the new device driver. A device driver essentially converts the more general input/output instructions of the operating system to messages that the device type can understand. A driver typically communicates with the device through the computer bus or communications subsystem to which the hardware connects. When a calling program invokes a routine in the driver, the driver issues commands to the device. Once the device sends data back to the driver, the driver may invoke routines in the original calling program. Drivers are hardware-dependent and operating-system-specific. They usually provide the interrupt handling required for any necessary asynchronous time-dependent hardware interface

Application Framework In computer programming, an application framework consists of a software framework used by software developers to implement the standard structure of an application. Application frameworks became popular with the rise of graphical user interfaces (GUIs), since these tended to promote a standard structure for applications. Programmers find it much simpler to create automatic GUI creation tools when using a standard framework, since this defines the underlying code structure of the application in advance. Developers usually use object-oriented programming techniques to implement frameworks such that the unique parts of an application can simply inherit from pre-existing classes in the framework.

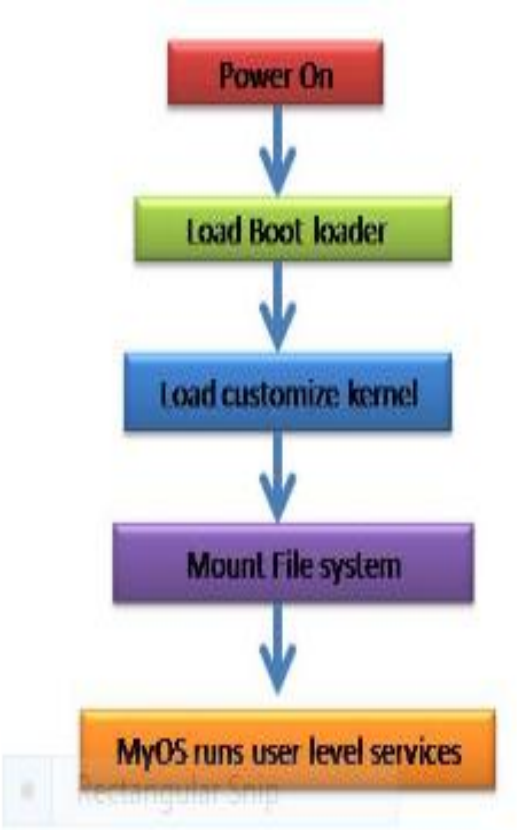


Fig 1.3: Smartphone Flow Chart of Software Start-up

1.3.1 Mobile Architecture V/S Computer Architecture

The basic architecture of the Smartphone and the desk top computer is similar in many ways. However, there are several dissimilarities between these architectures as given below.

S.N	Parameter	Desktop Computer	Mobile Device
1.	Processor	Intel, AMD (64 bit)	ARM/ Intel Atom (32 bit)
2.	Memory	Disk memory is mandatory	No disk memory
3.	OS	Desktop and non real time OS [One Desktop can have multiple operating System]	RTOS(Mobile RTOS) [One Mobile Device can have only one OS
4.	Protocols	GSM/3G/4G/CDMA not available	Mandatory: GSM/3G/4G/CDMA
5.	Booting Time	More	Less
6.	Power Consumption	High Power [in watts]	Low Power [in milli-watts]

1.4 SoC Architecture

System on Chip (SoC) is an integrated circuit(IC) that integrates all components of a computer, or other electronic system into a single chip. It may contain digital, analog, mixed-signal and radio frequency functions- all on a single chip substrate. SOC includes on chip memory (RAM & ROM), the microprocessors, peripheral interfaces, I/O logic control, data converters & other components that comprises a mobile device.

A System on Chip SoC's are found in most of the consumer products ranging from modems, DVD players, set top boxes, television and mobile phones etc. there exist several vendors such as TI, Intel, Qualcomm, Broadcom etc. produces SoCs like OMAP, Quark, snapdragon, BCM respectively. OMAP stands for Open Multimedia Applications Platform. OMAP is a series of System on Chips for portable and mobile multimedia applications. SoCs generally include a general-purpose ARM processor core plus one or more specialized co-processors. Snapdragon is a family of mobile systems on a chip by Qualcomm. Qualcomm considers Snapdragon a "platform" for use in smart phones, tablets and smart book devices.
When it is not feasible to construct a SoC for a particular application, an alternative is a System in Package (SiP) comprising a number of chips in a single package. In large volumes, SoC is believed to be more cost-effective than SiP since it increases the yield of the fabrication and because its packaging is simpler. The generic architecture of SoC is shown in figure 1.5.

- The SoC's meant for mobile devices should consist of:
- Support for internet browsing/email.
- Intense graphics and sound, powerful processing (GPU) for games, videos and music.
- Good Image Processing module for Camera and Videos
- GPS chip and compass for GPS
- GSM, 3G, Bluetooth communication support.
- Antenna and Power Control for general purpose.
- Inbuilt Memory (RAM & FLASH).

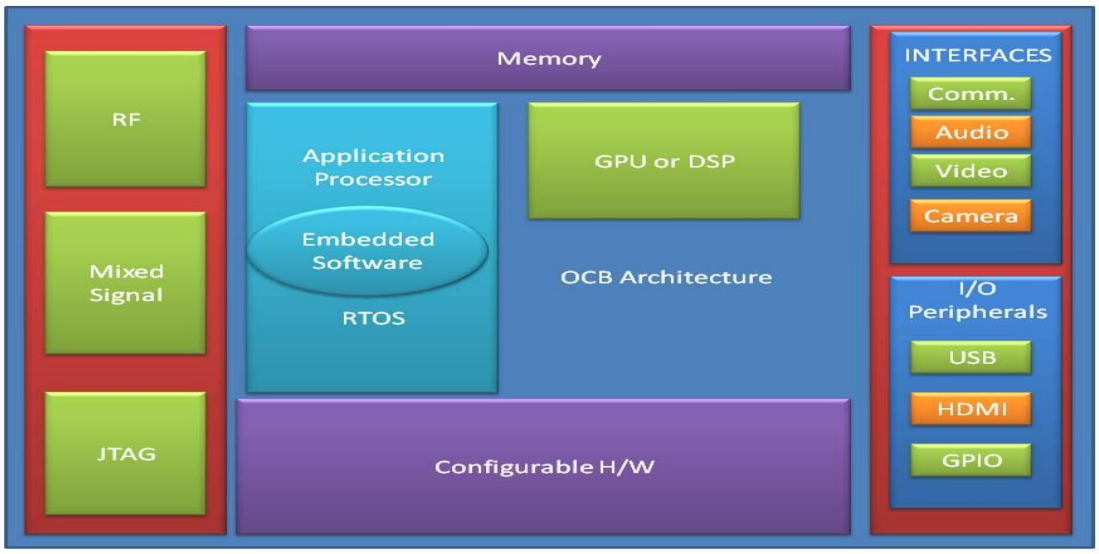


Fig 1.4: Generic SoC Architecture

1.4.1 Characteristics of SoC

The basic characteristics of mobile devices SoC's are:

1. Small Form Factor
2. High Performance
3. Low Power Consumption
4. Smaller space requirements
5. More memory supports
6. Analog mixed signal Integration
7. Higher system reliability & Low Cost
8. Mobile OS support

1.4.2 Merits and Demerits of SoC

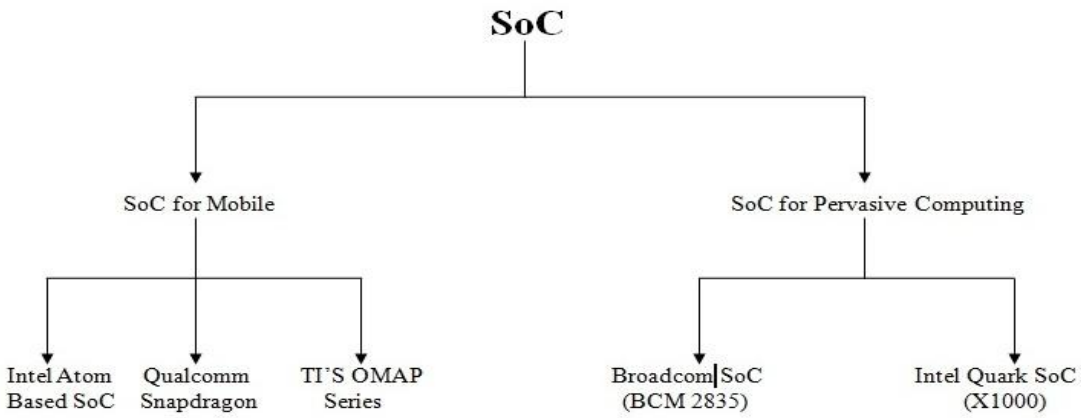
The **Merits** of SoC includes

1. It consumes less power. Around 90% is consumed by data bus and address bus cabling. Since in a SoC the components are internally connected with in a small size leads to less power consumption.
2. Smaller Board Space: Individual chip components if used result into heaviness and increased chip size. Integration of all chip components into one result in smaller size and weight.
3. All the components are on the same board and are internally connected; therefore a lot of cabling is saved in this respect which leads to lesser cabling cost.
4. Lower cost per gate due to advances in VLSI Technologies.
5. Greater design security at hardware and firmware level.
6. Faster execution due to high speed application processor & memory.
7. SoC based designs are more reliable and provides better performance.

The **Demerits** of SoC includes

1. Initial design and development cost is high. If number of units are small then cost per SoC is very high.
2. Even when a single system or transistor in SoC gets failed, the complete SoC has to be replaced. It results in Board Level Servicing which is costly.
3. Increased system complexity.
4. For high power applications SoC are not suitable.

1.4.3 Classification of SoC



1.4.3.1 SoC's for Mobile

There are various types of SoC's available for Mobile Device as follows:

a) Atom SoC

Atom is a system on chip (SoC) platform designed for smart phones and tablets launched by Intel in 2012. It is a continuation of the partnership announced by Intel and Google on 13 September 2011 to provide support for the Android operating system on Intel x86 processors. Atom range competes with existing SoCs developed for the Smartphone and tablet market from companies like Texas instruments, Nvidia, Qualcomm and Samsung. The Intel Atom processors Z2580 (2.0 GHz), Z2560 (1.6 GHz) and Z2520 (1.2 GHz) deliver dual core performance built on Intel's leading 32 nm process technology. Intel Hyper-Threading Technology1 supports four simultaneous application threads for smooth and seamless multitasking and responsive Web browsing. The Architecture of Atom SOC is shown in figure 1.6.

Characteristics of Atom SoC

i. Intel has adapted the x86 based Atom line CPU developed for low power usage for hand held and mobile devices.
ii. The SOC include the Intel Graphic Media Accelerator for compelling and realistic 3D gaming experiences, 1080pHD video, and crystal-clear graphics, WUXGA 1920 *1200 display support for the larger screens of tablets.
iii. Intel technology also includes dynamic frequency and dynamic voltage scaling for outstanding power efficiency.

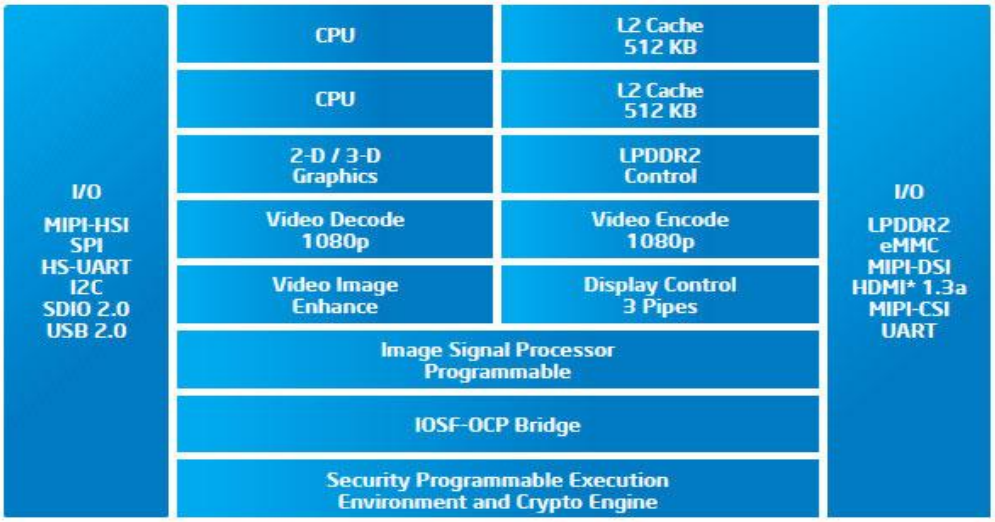


Fig 1.5: Block Diagram of Intel Atom SoC [1]

b) Qualcomm Snapdragon

Snapdragon is a family of mobile system on a chip (SoC) by Qualcomm. Qualcomm considers Snapdragon a "platform" for use in smart phones, tablets, and smartbook devices. The original Snapdragon CPU namely **SCORPION** is Qualcomm's own design similar to those of the ARM Cortex-A8 core and it is based on the ARMv7 instruction set. but it has much higher performance for multimedia-related SIMD operations. The successor to Scorpion, found in S4 Snapdragon SoCs, is named **Krait** CPU. It has many similarities with the ARM Cortex-A15 CPU and is based on the ARMv7 instruction set.

The majority of Snapdragon processors contain the circuitry to decode high-definition video (HD) resolution at 720p or 1080p depending on the Snapdragon chip. Adreno, the company's proprietary GPU series, integrated into Snapdragon chips (and certain other Qualcomm chips) is Qualcomm's own design, using assets the company acquired from AMD. The Adreno 225 GPU in Snapdragon S4 SoCs adds support for DirectX 9/Shader Model 3.0, which makes it compatible with Microsoft's Windows 8.

All Snapdragons feature one or more DSPs called Hexagon namely QDSP5 or QDSP6 in the modem and multimedia parts. In the Snapdragon 200–800 series, one of the multimedia Hexagons is programmable by the users through the Hexagon SDK. The multimedia Hexagons are mostly used for audio encoding/decoding, the newer Snapdragons have a hardware block called Venus for video encoding/decoding. The following table shows the comparisons of the different generations of Snapdragon.

Characteristics of Snapdragon SoC

i. Compared to SoC's from many competitors, Snapdragon SoC's have been unique in that they have the antenna for cellular communication on-die. That means they do not require a separate external antenna on the PCB.
ii. Snapdragon SoC also have on-die Wi-Fi, GPS/GLONASS and Bluetooth basebands.

iii. The integration of the above mentioned features reduces the complexity and cost of the final design for the OEM.
iv. It uses the 28 nm technology in most of the S4 SoCs.

	Snapdragon 800	Snapdragon 600	Snapdragon 400	Snapdragon 200
CPU	Up to 2.3 GHz Quad Krait 400 CPU	Up to 1.9 GHz Quad Krait 300 CPU	Up to 1.7 GHz Dual Krait 300 CPU	Up to 1.4 GHz Quad Cortex A5 CPU
GPU	Adreno 330 GPU	Adreno 320 GPU	Adreno 305 GPU	Adreno 203 GPU
DSP	Hexagon, QDSP6V5A, 600MHz	Hexagon, QDSP6V4, 500MHz	Hexagon, QDSP6V4, 500MHz	QDSP5, 384MHz
Video	4k x 2k UHD video capture/playback	1080p HD video	1080p HD video	720p HD Video (30/15 fps)
Modem	3G/4G World/multimode LTE on select processors	No modem	3G/4G World/multimode LTE on select processors	3G CDMA/UMTS/GSM on select processors
Camera	Up to 55MP, Stereoscopic 3D, Dual ISP	Up to 21MP, Stereoscopic 3D	Up to 13.5MP, Stereoscopic 3D on select processors	Up to 8MP
GPS	IZat Gen8B	IZat Gen8A	IZat Gen8A	IZat Gen7A
USB	USB 3.0/2.0	USB 2.0	USB 2.0	USB 2.0
Bluetooth	BT4.0 Integrated digital core	BT4.0 Integrated digital core	BT4.0 Integrated digital core	BT4.0 Integrated digital core
WiFi	802.11n/ac (2.4/5GHz) Integrated digital core	802.11n/ac (2.4/5GHz) Integrated digital core	802.11n/ac (2.4/5GHz) Integrated digital core	802.11n/ac (2.4/5GHz) Integrated digital core

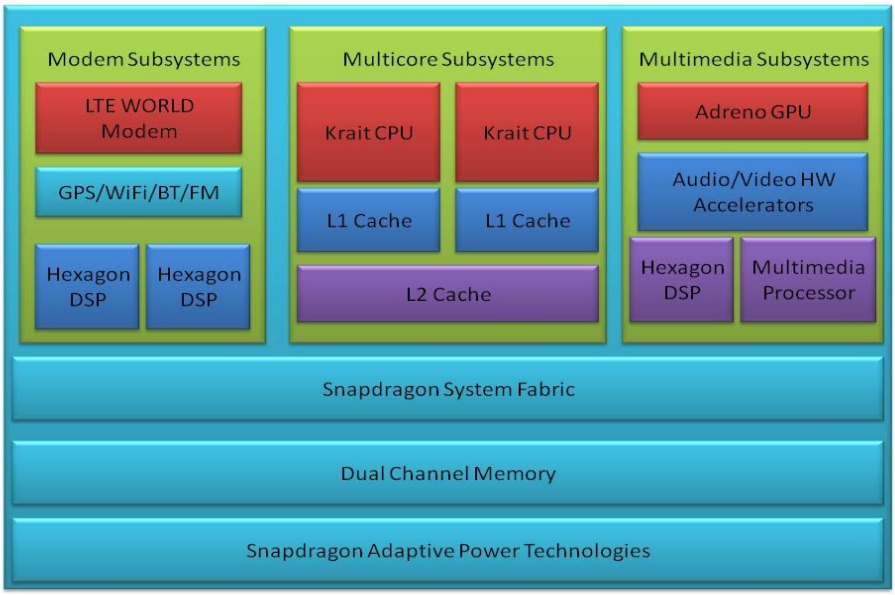


Fig 1.6: Block Diagram of Snapdragon SoC [2]

c) OMAP SoC

OMAP (Open Multimedia Applications Platform) is a series of image/video processors developed by Texas Instruments. They are a category of proprietary system on chips (SoCs) for portable and mobile multimedia applications. OMAP devices generally include a general-purpose ARM architecture processor core plus one or more specialized co-processors. Earlier OMAP variants commonly featured a variant of the Texas Instruments TMS320 series digital signal processor.

The OMAP family consists of three product groups classified by performance and intended application:

- High-performance applications processors
- Basic multimedia applications processors
- Integrated modem and applications processor

The Block diagram of OMAP4 is shown in figure 1.8. The various components and their communication interfaces with different processors shows the complexity of the SoC design. However it makes easy to prototype the smart phones and shorter time to market deadlines.

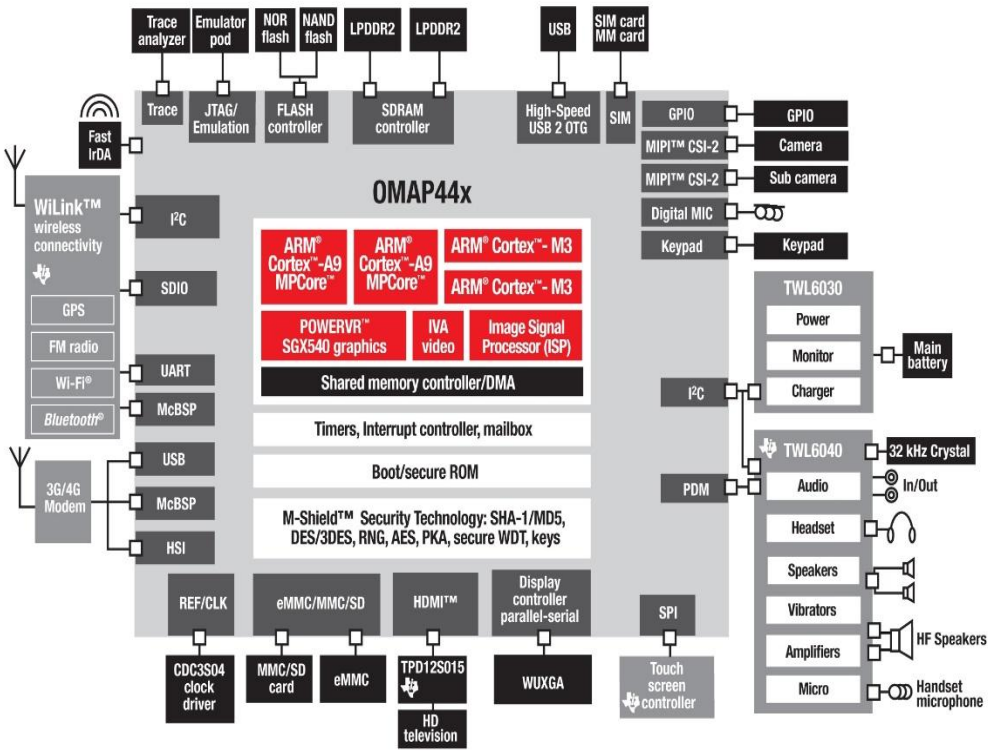


Fig 1.7: Block Diagram of OMAP SoC [3]

1.4.3.2 SoC for Pervasive Computing

The different SoCs available for pervasive computing are described as follows-

a) Broadcom BCM2836 SoC

The new Raspberry Pi 2 is the advance version of raspberri Pi1 with several new features. It has a quad core ARMv7 900MHz processor and with 1GB RAM. It's 6 times faster than Pi1 at the same cost of Pi1. It is a card sized Linux computer and supports android, windows10 operating systems. Its integrated with GPU Videocore IV 250 Mhz, a 40-pin GPIO, four USB ports and 10/100 thernet. Technical Specifications of Pi 2 aare as follows:

- SoC: Broadcom 2836
- CPU: Quad-core ARM7 900MHz
- GPU: Videocore IV 250MHz
- Memory: 1GB
- GPIO: 40pin
- Ports: 4x USB 2.0, 100BaseT Ethernet, HDMI, MicroSD card
- Size: 85.60 × 56.5mm (about 3.2 x 2.1-inch)

Other specifications:

- 4 x USB ports
- 4 pole Stereo output and Composite video port
- Full size HDMI
- CSI camera port for connecting the Raspberry Pi camera
- DSI display port for connecting the Raspberry Pi touch screen display
- Micro SD port for loading your operating system and storing data
- Micro USB power source

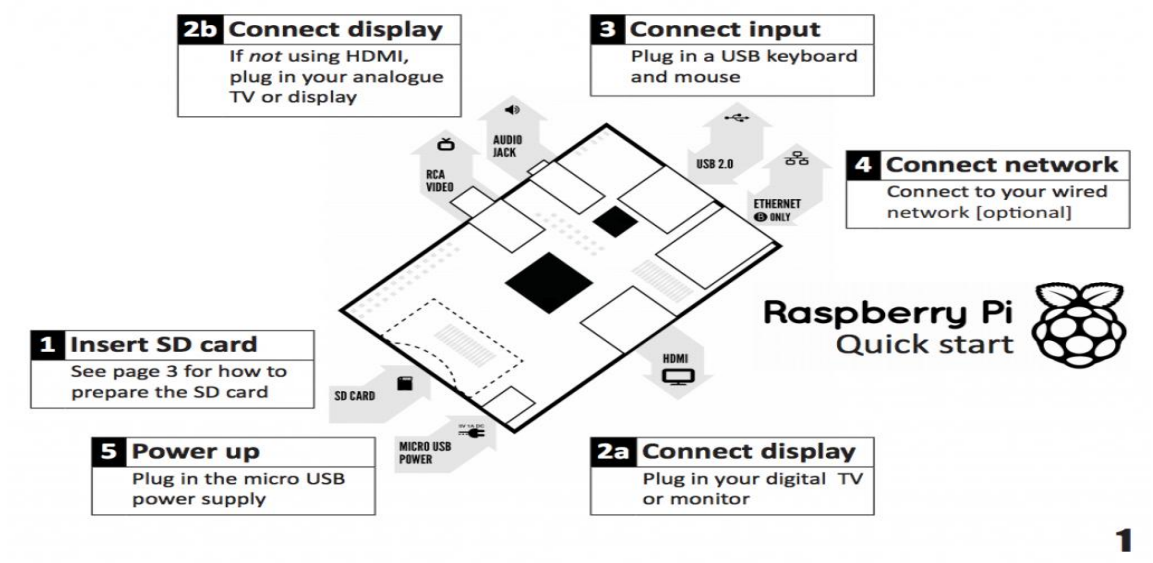


Fig: Block Diagram of BCM 2836 based RaspberriPi [4]

Characteristics of Broadcom BCM 2835

i. Low power ARM1176JZ-F Applications Processor.
ii. Dual Core VideoCore IV Multimedia Co-Processor.
iii. 1080p30 Full HD HP H.264 Video Encoder/Decoder advanced Image Sensor Pipeline (ISP) for up to 20 Mpixel cameras operating up to 220 Mpixel per second.
iv. Low power, high performance OpenGL-ES-1.1/2.0 VideoCore GPU.
v. High performance display outputs with simultaneous high resolution LCD and HDMI with HDCP at 1080p60.

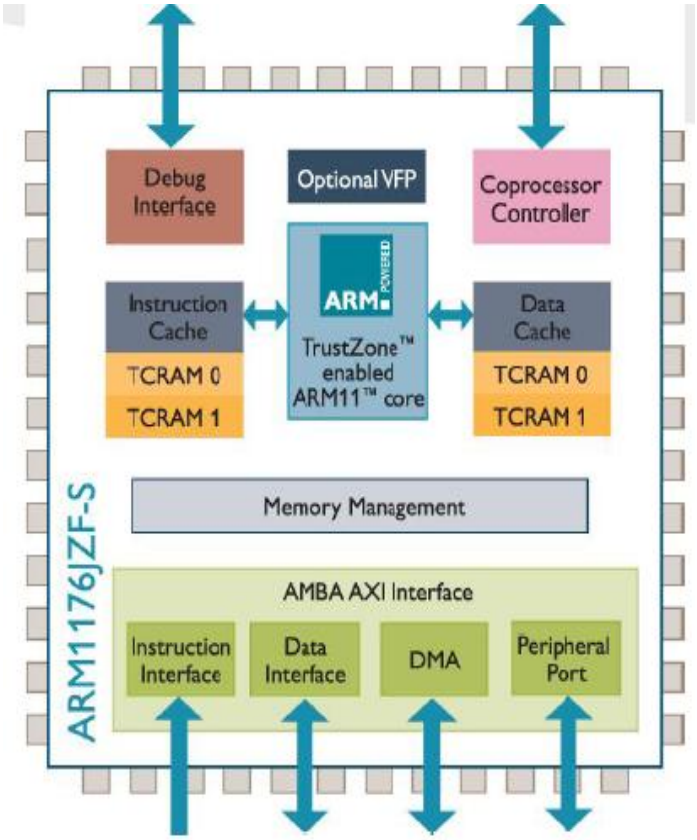


Fig 1.8: Block Diagram of Broadcom SoC (BCM2836) of Raspberry Pi [4]

1.4.4 SoC Selection

Broadcom BCM2836 SoC is selected in this book for design & development of Smartphone because of the following reasons:

a) **Power consumption** - The BCM2836 draws about five to seven watts of power.
b) **No moving parts**- The BCM2836 used in Raspberry Pi uses an SD card for storage, which is fast and has no moving parts. There are also no fans and other things to worry about. A Class 10 SD card is usually the best performing compared to lower class cards, but this will mainly only affect boot time where there is the most I/O.
c) **Small form factor** - The Pi (with a case) can be held in your hand.
d) **No noise** - . RPi does not produce any noise while running.
e) **Status lights** - There are several status lights on the Pi's motherboard. With a clear case you can see NIC activity, disk I/O, power status, etc.
f) **Expansion capabilities** - There are numerous devices available for the Pi, all at very affordable prices- everything from an I/O board (GPIO) to a camera. The Pi has two to four USB ports depending upon the model.
g) **Built-in HDMI capable graphics** - The display port on the Pi is HDMI and can handle resolutions up to 1920×1200. It leads to develop video player and run the gaming applications**.**
h) **Affordable** - Compared to other similar alternatives, the Pi (revision B) offers the best specs for the price. It is one of the few devices in its class that offers 512 MB to 1 GB of RAM. The Pi has come down in price of approximately Rs. 2500.
i) **Huge community support** - The Pi has phenomenal community support. Support can be obtained quite easily for the hardware and/or GNU/Linux software that runs on the Pi mainly in user forums, depending on the GNU/Linux distribution used.
j) OS Support - At presents it supports Linux OS. Android kernel is also ported into the Pi. Windows 10 is also accepted to support for Pi 2 for IoT applications.

1.5 Mobile Booting

Booting (also known as booting up) is the initial set of operations that a computer system performs after power to the processor is switched on or when the system is reset. The booting process begins when a system is turned on for the first time, or when the operator invokes a load function from the console, and ends when the system is ready to perform its normal operations and the control has been transferred to the user.

When a computer starts up the first thing that occurs is it send a signal to motherboard which in turn starts the power supply. After supplying the correct amount of power to each device, it sends a signal called "Power OK" to BIOS which resides on motherboard. Once the BIOS receive the "Power OK" signal, it starts the booting process by first initializing a process called POST (Power on self test). POST first check that every device has right amount of power and then it check whether the memory is not corrupted. It initializes each device and gives control to BIOS for further booting.

The BIOS first find 512 bytes of image called MBR (Master Boot Record) or Boot sector from the floppy disk or hard disk which is used for booting. The priority of boot devices is set by the user in BIOS setting. Once BIOS finds the boot sector it loads the image in memory and executes it. If a valid boot sector is not found, BIOS check for next drive in boot sequence until it find valid boot sector. It is boot sectors responsibility to load the operating system in memory and execute it as a result the control transfer to OS and further applications as required by the user.

1.5.1 General Booting Process in a Mobile Device

The booting process slightly differ and faster in the mobile device. the steps of booting in the mobile device is given in the figure 1.9.

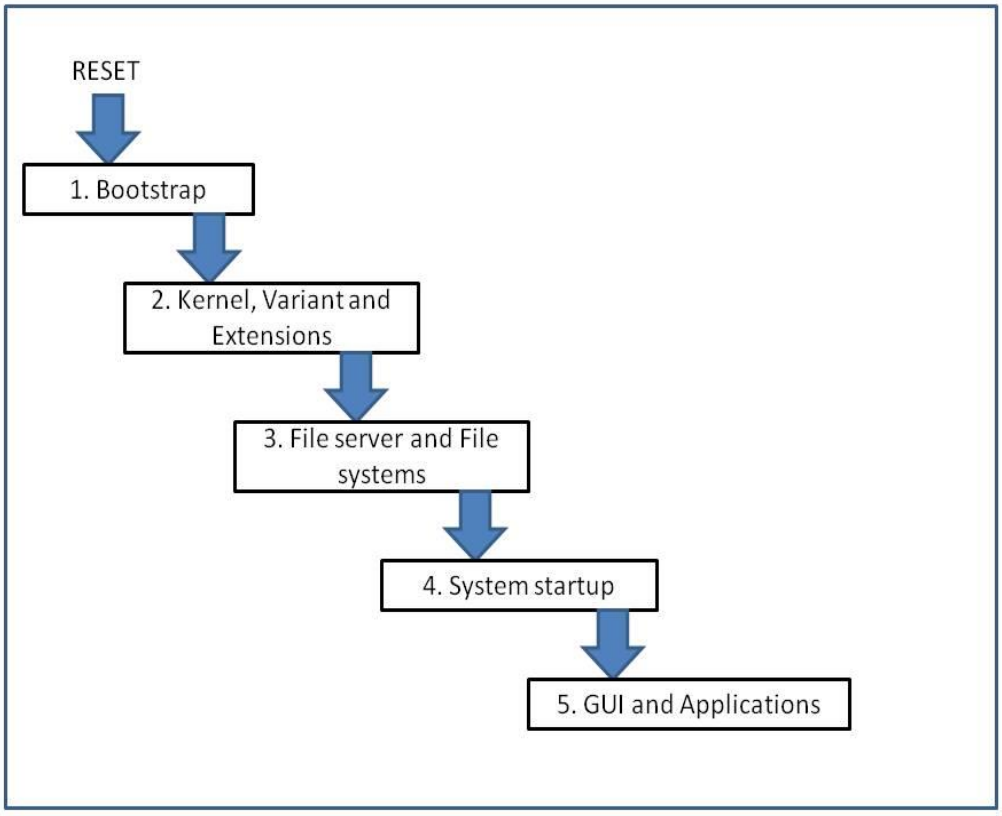


Fig 1.9: Booting Process in a mobile device

Bootstrap As soon as the power button is pressed, it resets the CPU (registers values are set to predefined value). The code which executes the task at the initial stages of booting is hardwired and is stored at a fixed location in a ROM. As the hardware connected to mobile is always fixed, so the preliminary task is very basic which includes checking of all chip components working properly. The boot loader loads the kernel (Mobile OS) into RAM. Then next is the job of boot loader, whose job is to locate and load the kernel.

Kernel Board Support Packages specific to the mobile device is executed. It is responsible for the assigning of the various functionalities to the controllers and pins of the chip of the mobile device. After this kernel operations such as initialization of interrupt controllers, setting up of memory protections, caches and scheduling, memory management functions are executed.

File system File system is way of organizing data. This stage is loaded by boot loader and is called in the middle of Kernel stage. All the data present in the mobile or any system is organized, stored and updated in the mobile device.

System Startup As the kernel and file system are up, then the system startup process can begin. From this stage all the processes that occur are related directly with the user space applications. Several preinstalled applications such as Web manager, Sound manager, Graphics manager etc. are brought into main memory and other default applications are launched.

GUI Graphical user interface is the final stage of mobile booting process, which is seen by everyone at the end of booting process. This leads to the display of the user defined settings, application launching through which user interacts with the mobile.

References

[1]http://hothardware.com/news/Intel-Launches-Clover-Trail-Atom-Z2760-SoC-and-Multiple-Windows-8-Tablet-Design-Wins.
[2]https://eda360insider.wordpress.com/2011/10/10/qualcomm-reveals-more-snapdragon-4-soc-details-in-a-white-paper-want-to-know-what%E2%80%99s-inside/.
[3]http://www.ti.com/general/docs/wtbu/wtbuproductcontent.tsp?contentId=53243&navigationId=12843&templateId=6123.
[4]http://www.petervis.com/Raspberry_PI/BCM2836_Block_Diagram/BCM2836_Block_Diagram.html.

Chapter 2

Introduction to Mobile Programming

2.1 Introduction

Mobile programming is the set of processes and procedures involved in writing software for portable, wireless computing devices such as smartphones or tablets. It involves the development of various software applications for Smartphone such as gaming, educational, multimedia and business applications. Software development process of any mobile or smart phone is similar to web application development in some sense and has its root in more traditional software development. But there are some crucial differences like the mobile applications (apps) are often written specifically to take advantage of the unique features a particular mobile device offers specific to mobile operating system. For example, a gaming application might be written to take advantage of the accelerometer in built in the mobile phone of particular operating system. Similarly a sensor that is provided in the device or sensors API's that are provided in the SDK of the particular operating system helps to develop applications. The developed applications may be pre-installed on mobile phones during manufacturing process, or may be delivered as web applications downloadable from the manufacturers' server-side (mobile app store). Several SDK's helps the user's to create applications and install the same in the target devices e.g. Android SDK.

To develop any software application either for a mobile or desktop, the most crucial part is the selection of programming language for a specific mobile operating system, which if not judicially selected may lead to the failure of an application. Each programming language has its syntax, code structure and dependent libraries that differs from others. Several programming languages such as Java, C, python, objective C etc. are being used for development of mobile application.

2.2 Programming Languages for Mobile Application Development

Different programming languages used for mobile application development. The brief description of these languages are given below:

- **Java: -** It is an object-oriented platform independent programming language. Different versions of java are available. J2ME is specially used for mobile application development. Java supports several operating system. Java is widely used in Android applications.
- **C++: -** C++ is best suited for low-level programming. It works extremely fast and is used to develop high-performance native applications e.g. in android operating system.
- **Objective-C: -** It is a programming language used for writing software exclusively with native iOS platforms and Apple's OS X. Objective-C is a superset of the C programming language, having small, easy to learn and object-oriented syntax.
- **HTML5**: -HTML5 is a revision of the Hypertext Markup Language (HTML), the standard programming language for describing the contents and appearance of Web pages.
- **Python:** -Python [1] is a high-level, interpreted, interactive and object-oriented scripting language. It was designed to be highly readable which uses English keywords frequently where as other languages use punctuation and it has fewer

syntactical constructions than other languages. It mainly used for prototyping initial applications and ease to use and learn. It also supports cross platforms and completely open source.

2.3 Various Programming Languages Vs OS platforms

Various programming languages that supports for different mobile operating systems are given in table 2.1.

S.No.	Mobile OS	Supported Programming Language
1	Android	Java but portions of code can be in C, C++
2	Blackberry	C++
3	Java ME	Java
4	iOS	Objective C
5	Windows Phone	C#, .NET

Table.2.1: OS & supported Programming Languages

2.4 Python as a programming language

In this book, Python is being selected as a mobile programming language for development of applications due to following reasons:

- **Beginner's Language**: Python is a language for the beginners. It supports the development of a wide range of applications from simple text processing such as web browsers, games, etc and is widely used for research and development across different domains.
- **Easy-to-read, learn, and maintain:** Python has relatively few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language in a relatively short period. It is easy to read, learn and maintain.
- **Portable & Extendable:** Python can run on a wide variety of hardware platforms with the same interface. These platforms includes ARM, INTEL, etc. Programmer can add low-level modules to the Python interpreter.
- **Interactive:** Interact with the interpreter directly to write programs at Python prompt.
- **Interpreted:** Program written in Python is processed at runtime by the interpreter and hence, compilation of programs before execution is not needed as required in PERL and PHP.
- **Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **A broad standard library:** One of Python's greatest strengths is its bulk of library. It is portable and cross-platform compatible on different OS such as UNIX, Windows and Macintosh.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh and the X Window system of UNIX.
- **Databases:** Python provides interfaces to all major commercial databases.
- **Callable:** Python provides a better structure and support for large programs than shell scripting.

2.5 Python Installation

Python distribution is available for a wide variety of platforms. Installation of Python in any computing device is very simple that includes the downloading of binary code applicable for specific and user defined platform followed by installation. If the binary code for defined platform is unavailable, C compiler can be used to compile the source code manually. Compilation of the source code offers more flexibility in terms of choice of features that are required by the programmer in installation. The python installation process is different for different OS platform which is described in step by step manner in following sub section.

2.5.1 Procedure of Python Installation on Unix & Linux Platforms

Following are the steps for installation of Python on Unix/Linux machine:

- Follow the link http://www.python.org/download/ to download zipped (compressed) source code available for Unix/Linux.
- The downloaded file is needed to be extracted for installation and for the same open a terminal on the screen and type the command (For tar.gz file) "tar –xvf rootfs.tar.gz – file_name" for unzipping of the downloaded file.
- Programmer may customize few options (if required) by editing the Modules/Setup file.
- Now Run ./configure script
- Run make command on terminal followed by make install to complete the installation process.

The above steps will install python in a standard location */usr/local/bin* and its libraries are installed in*/usr/local/lib/pythonXX* where XX is the version of Python installed.

2.5.2 Procedure of Python Installation on Windows Platforms

Steps for installation of Python on Windows platform are different from the steps followed for Linux/Unix platform. The steps are as given below:

- Open a Web browser and go to http://www.python.org/download/k for the Windows installer *python-XYZ.msi* file where XYZ is the version you are going to install.
- To use this installer *python-XYZ.msi*, the Windows system must support Microsoft Installer 2.0. Just save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file by double-clicking it in Windows Explorer. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the installation is finished, and you're ready to roll!.

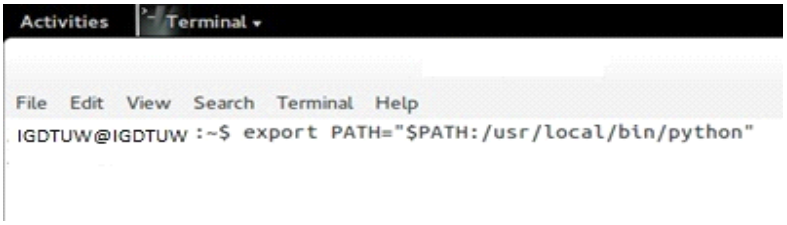
2.6 Setting up the execution PATH

Programs and other executable files can be present in many directories, so operating systems are provided a search path that lists the directories where the OS can search for executables. The path is stored in an environment variable, which is a named string maintained by the operating system. These variables contain information available to the command shell and other programs. The path variable is named PATH in Unix or Path in Windows (Unix is case-sensitive, Windows is not).

2.6.1 Setting path at Unix/Linux Platform

To add the Python directory to the path for a particular session in Unix:

- **In the csh shell:** type setenv PATH "$PATH:/usr/local/bin/python" and press Enter.
- **In the bash shell (Linux):**type export PATH="$PATH:/usr/local/bin/python" and press Enter.
- **In the sh or ksh shell:** type PATH="$PATH:/usr/local/bin/python" and press Enter.



Note: /usr/local/bin/python is the path of the Python directory

2.6.2 Setting path at Windows

To add the Python directory to the path for a particular session in Windows:

- Open command window and type- **path %path%;C:\Python** followed by pressing of Enter key on keyboard.

2.7 Running a python script

A Python script can be executed at command line by invoking the interpreter on your application, as in the following: **$ python script.py.** All python files will have extension "**.py"**. Save the following source code in a filename.py (eg. igdtuw.py) file

Example -1 Hello World Program

- Open a text file and write a code.

  Print "Hello World!"
- Save the text file as filename.py (eg. igdtuw.py)
- Running a code by : $ sudo python igdtuw.py

Output of the program is Hello World

2.8 Basic Python Concepts

- **The print function: -** The print function simply outputs a string value to our script's console. It's very useful for communicating with users or outputting important information. You can use double quotes or single quotes. For multi-line strings you can use triple quotes. Syntax is:

```
print "IGDTUW"
print 'Hello World!'
print """
This is line one.
This is line two.
This is line three.
Etc…
      """
```

- **Reading Keyboard Input: -** Python provides two built-in functions to read a line of text from standard input, which by default comes from the keyboard.

i. ***The raw-input Function:*** The *raw_input([prompt])* function reads one line from standard input and returns it as a string (removing the trailing newline).

```
# it will return the string value

str = raw_input("Enter your input: ");
print "Received input is : ", str
```

ii. ***The input Function:*** The *input([prompt])* function is equivalent to raw_input, except that it assumes the input is a valid Python expression and returns the evaluated result to you.

```
# it will return the integer value

str = input("Enter your input: ");
print "Received input is : ", str
```

- **Adding Comments: -** Comments allow us to add useful information to our scripts, which the Python interpreter will ignore completely. Each line for a comment must begin with the number sign '#'.

```
# This is a python program to display.

print "IGDTUW"
```

- **Strings and Control Characters: -** There are several control characters which allow us to modify or change the way character strings get printed by the print function. They're composed of a backslash followed by a character. Here are a few examples:
  \n : New line , \t : Tabs , \\ : Backslash , \' : Single Quote , \" : Double Quote

- **Variables: -** Like all programming languages, Python variables are similar to variables in Algebra. They act as place holders or symbolic representations for a value, which may or may not change over time. Here are six of the most important variable types in Python:
  int Plain Integers (25) , long Long Integers (4294967296) , float Floating-point Numbers(3.14159265358979) , bool Booleans (True, False), str Strings ("Hello World!") list Sequenced List ([25, 50, 75, 100]).

- **Rules for Naming Variables: -** You can use letters, digits, and underscores when naming your variables. But, you cannot start with a digit.
  var = 0 # Ok. **,** var1 = 0 # Ok. **,** var_1= 0 # Ok. **,** _var = 0 # Ok. **,** 1var = 0 # Syntax Error!

- **Arithmetic Operators: -** Arithmetic Operators allow us to perform mathematical operations on two variables or values. Each operator returns the result of the specified operation.

Operator	Name	Explanation	Examples
+	Plus	Adds the two objects	3+5 gives 8. 'a' +'b' give ab
-	Minus	Either give the negative number or gives the subtraction of one number from the other	-5.2 gives a negative number. 50-24 gives 26
*	Multiply	Gives the multiplication of two numbers or returns the string repeated that many times	2 * 3 gives 6. 'la' * 3 gives lalala
**	Power	Return x to the power of y	3 ** 4 gives 81(i.e. 3*3*3*)
/	Divide	Divide x by y	4/3 gives 1 (division of integers gives an integer). 4.0/3 or 4/3.0 gives 1.3333333333

- **Comparison Operators: -** Comparison Operators return a True or False value for the two variables or values being compared.

Operator	Purpose	Example
<	Returns True if the first value is less than the second value.	Value1 < Value2
<=	Returns True if the first value is less than or equal to the second value.	Value1 <= Value2
>	Returns True if the first value is greater than the second value.	Value1 > Value2
>=	Returns True if the first value is greater than or equal to the second value.	Value1 >= Value2
=	Returns True if the first value is equal to the second value.	Value1 = Value2
<>	Returns True if the first value is not equal to the second value.	Value1 <> Value2

- **Type Conversion: -** The special constructor functions int, long, float, complex, and bool can be used to produce numbers of a specific type. For example: if you have a variable that is being used as a float, but you want to use it like an integer do this: myFloat = 25.12, myInt = 25 , print myInt + int(myFloat)

- **Flow Control: -** Flow Control allows a program or script to alter its flow of execution based on some condition or test. The most important keywords for performing Flow Control in Python are if, else, elif, for, and while.

If Statement: The most basic form of Flow Control is the if statement.

```
# If the student's marks are less than or equal to 40 – Fail him!

if marks <= 40:
print "You're Fail!"
```

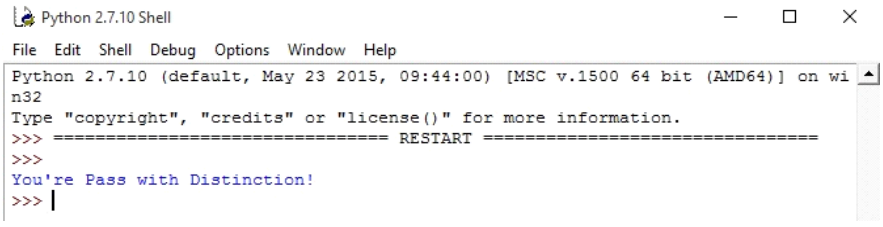
If-else Statement: The if-else statement allows us to pick one of two possible actions instead of all-or-nothing choice.

```
# If the student's marks are less than or equal to 40 – Fail him else Pass him!

marks = 75
if marks <= 40:
   print "you're Fail!"
else:
   print "you're Pass!"
```

If-elif-else Statement: The if-elif-else statement allows us to pick one of several possible actions by chaining two or more if statements together.

```
# If the student's marks are less than or equal to 40 – Fail him else if marks are less than 75- Pass him but not with distinction else Pass him with distinction!

marks = 80
if marks <= 40:
   print "You're Fail!"
elif marks < 75:
   print "You're Pass - but not with distinction!"
else:
   print "You're Pass with Distinction!"
```

Output:



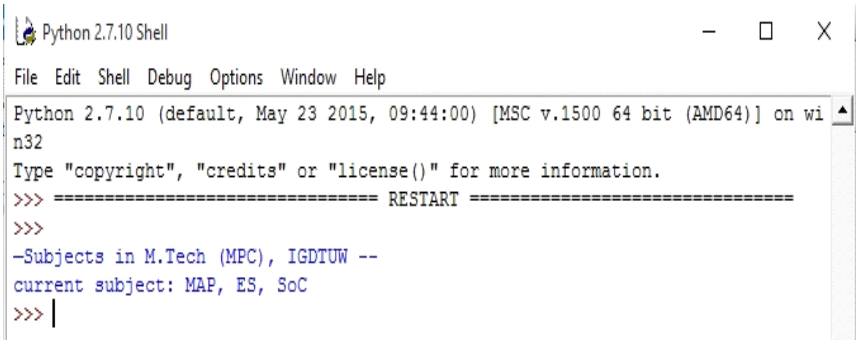
while Statement: The while statement allows us to continuously repeat an action until some condition is satisfied.

```
# Print value of count till count number 8 and then it will print IGDTUW

count = 0
while (count < 9):
   print 'The count is:', count
   count = count +
print "IGDTUW!"
```

for Statement: for statement allows to repeat an action based on the iteration of sequenced list.

```
# Print all subjects of M.Tech (MPC), IGDTUW given in array subjects.

subjects= ["MAP, ES, SoC"]
print "—Subjects in M.Tech (MPC), IGDTUW --"
for x in subjects:
   print 'current subject:', x
```

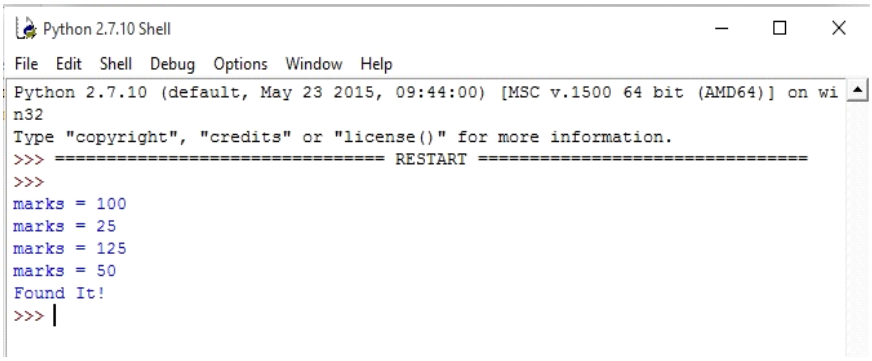
Output:



break Keyword: **-** The break keyword can be used to escape from while and for loops early.

```
# Print all the marks given in an array marks and as soon as it finds 50 marks it will stop the search.

marks = [100, 25, 125, 50, 150, 75, 175]
for x in marks:
   print 'marks =', x
   if x == 50:
      print "Found It!"
      break;
```

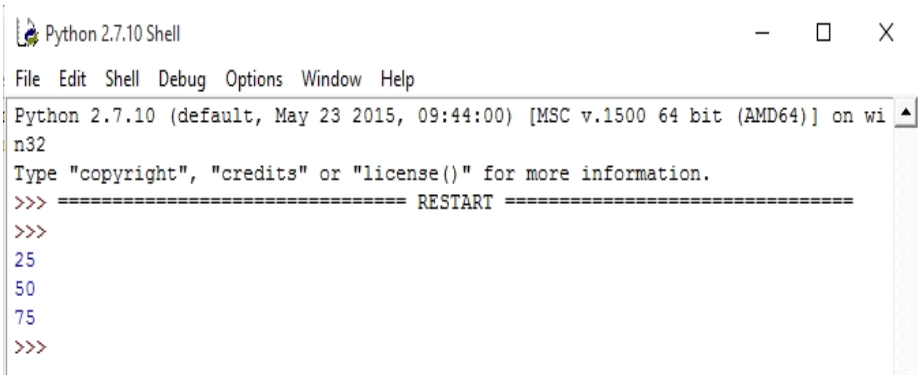
Output:



***continue Keyword:* -** The continue keyword can be used to short-circuit or bypass parts of a while or for loop.

```
# It will print all the marks in an array marks and will skip all triple digit numbers.

marks = [100, 25, 125, 50, 150, 75, 175]
for x in marks:
   if x >= 100:
      continue;
   print x
```

Output:



- **Functions: -** A function allows several Python statements to be grouped together so they can be called or executed repeatedly from somewhere else in the script. We use the def keyword to define a new function. Below, we define a new function called "printMarks", which simply prints out, "MAP Marks!" .

```
# On calling the function printMarks()- print MAP Marks

def printMarks():
     print "MAP Marks!"
```

- ***Function Argument:*** Often functions are required to perform some task based on information passed in by the user. These bits of Information are passed in using function arguments. Function arguments are defined within the parentheses "()", which are placed at the end of the function's name. New version of printMarks, can now print out customizable, "Marks of Mobile Architecture & Programming!", messages by using our new argument called "MAP".

```
# On calling the function printMarks()- print MAP Marks

def printMarks(MAP ):
print "Marks of Mobile Architecture & Programming " + MAP
+ "!"
```

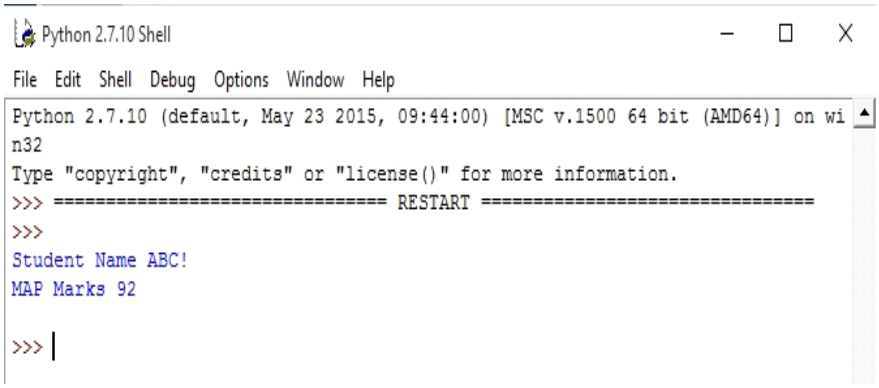
- ***Default Argument:*** Below, the argument called "MAP" has been assigned the default value of, "98".

```
# On calling the function printMarks()- print Marks of Mobile of Architecture & Programming 98

def printMarks( MAP="98" ):
print "Marks of Mobile Architecture & Programming " + MAP + "!"
```

- ***Keyword Arguments:*** If you really want to change the order of how the arguments get passed, you can use keyword arguments.

```
# On calling the function printName()- print student name and total marks of the student as passed through arguments.

def printName( studentname, totalmarks ):
    print "Student Name " + studentname + "!"
    print "MAP Marks " + str( totalmarks ) + "\n"
printName (totalmarks=92, studentname="ABC")
```

Output:



- **Basics of python GUI programming**

Python provides various tools/packages for developing graphical user interfaces (GUIs) such as Tkinter, wxPython, JPython.

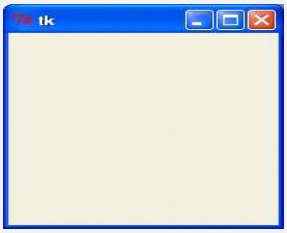
Tkinter: - Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. This tool is being used in this chapter for UI design & development.
wxPython: - This is an open-source Python interface for wxWindows http://wxpython.org.
JPython: - JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine http://www.jython.org.

- **Tkinter Programming:** - Tkinter is the standard GUI library for Python. Python with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. The step by step procedure us as follows:

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add the required widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Example:-

# creating a window where widgets will be added #!/usr/bin/python import Tkinter top = Tkinter.Tk() top.mainloop()	Output Window: 

- **Basic GUI elements (widgets):-** Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets i.e. GUI elements. Different GUI elements as well as their brief description is presented in the following table.

Operator	Description
Button	The Button widget is used to display buttons in your application.
Canvas	The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
Checkbutton	The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
Entry	The Entry widget is used to display a single-line text field for accepting values from a user.
Frame	The Frame widget is used as a container widget to organize other widgets.
Label	The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
Listbox	The Listbox widget is used to provide a list of options to a user.
Menubutton	The Menubutton widget is used to display menus in your application.
Menu	The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
Message	The Message widget is used to display multiline text fields for accepting values from a user.
Radiobutton	The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.
Scale	The Scale widget is used to provide a slider widget.
Scrollbar	The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.
Text	The Text widget is used to display text in multiple lines.

Toplevel	The Toplevel widget is used to provide a separate window container.
Spinbox	The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.
PanedWindow	A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
LabelFrame	A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
tkMessageBox	This module is used to display message boxes in your applications.

- **Programming with Basic GUI Elements :-** Brief description and programming of some basic GUI elements like Button, Frame, Checkbutton etc is presented in the following section.

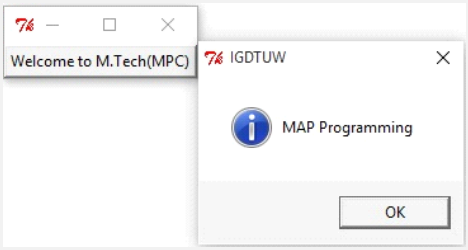
  Button: The Button widget is used to add buttons in a Python application. These buttons can display text or images that convey the purpose of the buttons. Syntax is:

  w = Button (master, option=value, ...)

Example:-

```
import Tkinter
import tkMessageBox

top = Tkinter.Tk()
def helloCallBack():
   tkMessageBox.showinfo(
"IGDTUW", "MAP Programming")

B = Tkinter.Button(top, text ="Welcome
to M.Tech(MPC)", command =
helloCallBack)

B.pack()
top.mainloop()
```

Output:



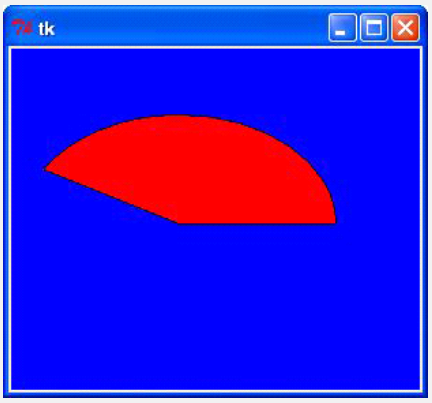
Canvas: The Canvas is a rectangular area intended for drawing pictures or other complex layouts. You can place graphics, text, widgets or frames on a Canvas. Syntax is

w = Canvas (master, option=value, ...)

Example:-

```
import Tkinter
import tkMessageBox

top = Tkinter.Tk()

C = Tkinter.Canvas(top, bg="blue", height=250, width=300)

coord = 10, 50, 240, 210
arc = C.create_arc(coord, start=0, extent=150, fill="red")

C.pack()
top.mainloop()
```

Output:



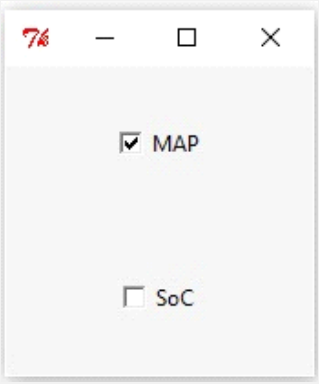
Checkbutton: The Checkbutton widget is used to display a number of options to a user as toggle buttons. The user can then select one or more options by clicking the button corresponding to each option. Syntax is:

w = Checkbutton (master, option, ...)

Example:-

```
from Tkinter import *
import tkMessageBox
import Tkinter

top = Tkinter.Tk()
CheckVar1 = IntVar()
CheckVar2 = IntVar()
C1 = Checkbutton(top, text = "MAP", variable = CheckVar1, \
                 onvalue = 1, offvalue = 0, height=5, \
                 width = 20)
C2 = Checkbutton(top, text = "SoC", variable = CheckVar2, \
                 onvalue = 1, offvalue = 0, height=5, \
                 width = 20)
C1.pack()
C2.pack()
top.mainloop()
```

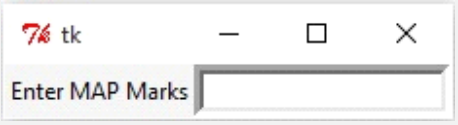
Output:



Entry: The Entry widget is used to accept single-line text strings from a user. Syntax is

w = Entry(master, option, ...)

Example:-

from Tkinter import * top = Tk() L1 = Label(top, text="User Name") L1.pack(side = LEFT) E1 = Entry(top, bd =5) E1.pack(side = RIGHT) top.mainloop()	**Output:** 

Exercise:

1. **Write a program to print factorial of a number using function.**
2. **Write a program to check if a string (e.g. MALAYALAM) is palindrome or not.**
3. **Write a program to extract a particular string from the file created by the user.**
4. **Write a program to create a database of Employee and use the CRUD operation.**

References:

1. http://www.tutorialspoint.com/python/python_overview.htm

Chapter 3

Operating Systems for Mobile Devices

3.1. Introduction

Operating system (OS) for mobile devices such as smartphones, tablets, PDAs etc. is known as mobile OS. Many computing devices like laptops are mobile in nature but their OS need not be considered as a mobile OS. This distinction is getting blurred in some newer operating systems that are hybrid and support for both. Examples of desktop operating systems are Windows, Mac OS X, Linux; and mobile operating systems are windows phone, android, iOS. Generally, OS provides services like file management, memory management, device management, process management etc. They also allow the users to install and run programs written for the concerned operating system. While Windows and Linux can be installed on standard PC hardware, Mac OS X can only run on Macintosh computers. Similar to desktop OS, each mobile OS manage files and folders, memory, processes etc. on the device. Mobile OS also facilitates the user to install and run applications written for the operating system. Inspite of many similarities between desktop OS and mobile OS both have following dissimilarities mentioned in table 3.1.

Parameters	**Desktop OS**	**Mobile OS**
Functionality	Designed to run on PC	Designed to run on mobile or hand held devices
User Support	Multi User Support	Single User Support
Real-time	Non real-time	Real-time (Hard)
Language support	C, C++, java, C#	Java, objective C, .NET
Memory RAM	RAM up to 16 GB	RAM up to 4 GB
Scheduling policy	Generally, RR and FCFS	Priority and RR
Power Optimization	Supports/Optional	Supports/Mandatory & Several techniques are being used
Memory Optimization	Supports	Mandatory/Several techniques are being used
Examples	Unix, DOS, Windows Mac OS	Windows Phone, iOS, Android

Table 3.1 Desktop OS Vs Mobile OS

Mobile OS combines the features of Desktop OS and features of mobile or hand held devices such as touch screen, cellular system, Bluetooth, Wi-Fi, GPS, camera, video, speech recognition, voice recorder, music player, near field communication etc. Operating system is an important aspect of mobile application development. It also plays a role in the development of tools, pre-packaged software, and device support. Various mobile operating system are discussed briefly in the next section.

3.2 Case study of Mobile OS:

Android: Android (based on the Linux Kernel) is from Google Inc. It has the largest installed base worldwide on smartphones. Most of Android is free and open source, but a large amount of software on Android devices (such as Play Store, Google Search, Google Play Services, Google Music, and so on) are proprietary ,licensed and closed source applications.

iOS: iOS is from .Apple Inc. It has second largest installed base worldwide on smartphones behind Android. It is closed source, proprietary and built on an open source Darwin core OS. The Apple iPhone, iPod Touch, iPad and second-generation Apple TVs uses iOS, which is derived from Mac OS X.

Windows Phone: Windows Phone is from Microsoft. It is closed source and proprietary. It has third largest installed base on smartphones behind Android and iOS. On February 15, 2010, Microsoft unveiled its next-generation mobile OS- Windows Phone. The new mobile OS includes a completely new UI inspired by Microsoft's "Metro_Design_Language". It not only includes full integration of Microsoft services such as OneDrive and Office, Xbox Music, Xbox Video, Xbox Live games and Bing, but also integrates many other non-Microsoft services such as Facebook and Google accounts. It is derived from Window OS.

Parameters	Android	Windows	iOS
Latest version	5.0 lollipop	Windows phone 8.1	iOS 8.4
Cost	Low cost	Relatively low cost	Expensive
Programming Language	C, C++, Java	C++, c#, Microsoft Visual Basic, JavaScript	Java
SDK platform	Linux, Mac OSX, windows	Mac OSX	Mac OSX via iOS SDK
Licence	Free and open source	Proprietary	Proprietary
CPU Architecture	ARM, MIPS, x86	ARM	ARM
Battery Life and management	Battery backup is good with battery saver options	Windows hone has a battery saver option that shows remaining. battery and allows you to turn off background apps	Apple has introduced detailed battery usage by app in iOS 8 but lacks battery saving mode

3.3 Customizing Linux OS for Raspberry Pi

Linux kernel is an open source, most flexible operating system that has ever been created. It supports a wide range of hardware platform and programming languages. Linux kernel customization is a process that makes the kernel faster and smaller compatible for specific hardware and application. Linux distribution provides the kernel source code but not exact as per user requirements. Several linear distributions are debian, wheezy, noobs, android etc.

The kernel is one component of a Linux OS and requires several libraries and applications to support end user's requirements. Linux supports various features such as portability, hardware support, scalability, exhaustive networking support, security, stability and reliability etc. Role of Linux OS is to manage all hardware and software resources and provide a set of APIs to support various hardware and applications.

Several customized Linux operating systems are ported on ubiquitous devices such as set up boxes, LCD TVs, wrist watches, PDA, gaming and medical and Mobile Devices.

3.3.1 Linux Source code and tool chain

The source code of Linux has grown to large size. Different parts of the kernel can be found in different directories. In the Linux system, the sources can normally be found under /usr/src/ Linux subdirectory. The exact directory structure of Linux is shown in Figure 3.1. The description of main directory /folder as describe in the table 3.1 Architecture-dependent code is held in the subdirectories of **arch/.** For example, **arch/arm/** for the Broadcom bcm2709 raspberry pi, **arch/1386/** for the Intel 386 and compatible processors.

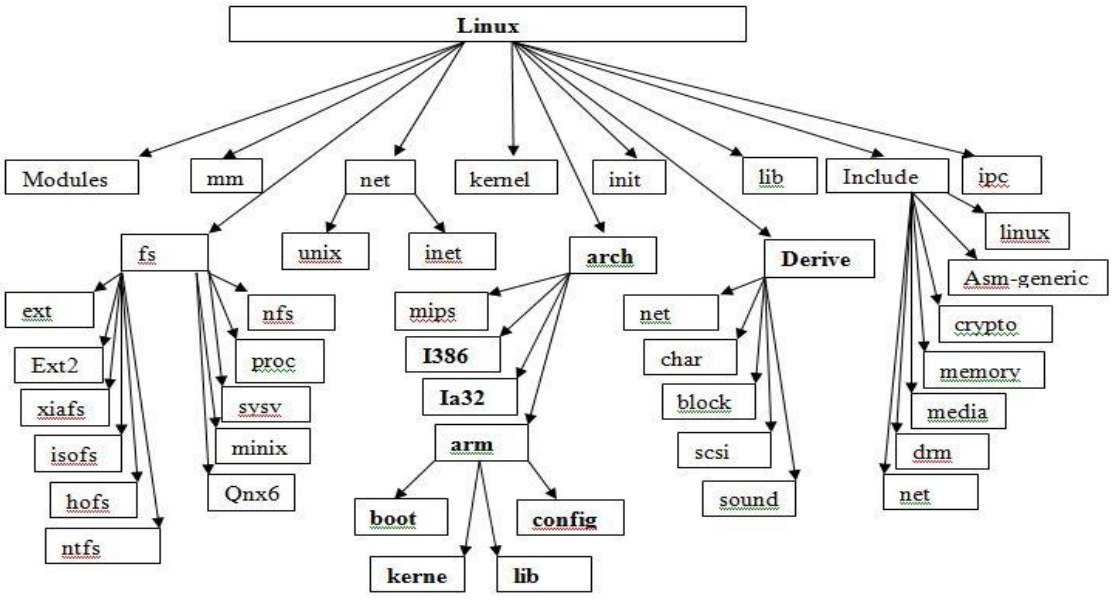


Figure 3.1 Directory structure of Linux OS

older	Description
Arch/<ARCH>	Architecture specific code
arch/<ARCH>/mach-<machine>	machine/board specific code
arch/<ARCH>/include/asm	architecture-specific headers
arch/<ARCH>/boot/dts	Device Tree source file, for some architectures
drivers/	All device drivers except sound ones (usb, pci...)
Kbuild	Part of the kernel build system
Kconfig	Top level description file for configuration parameters
Makefile	Top Linux Makefile (sets arch and version)
init/	Linux initialization (including main.c)
drivers/block/	the device drivers for block-oriented hardware (such as hard disks)

Table 3.1 Source Code Description

The drivers listed here are partially architecture-dependent and would properly belong to the **arch/*/** directory, where in **arch/arm/configs/** -contain the defined configuration file for specific hardware board. We just call the defconfig file at the time of compilation to set the compiler for desired configuration. The Linux customization is described below.

Step 1 Selection of Tool Chain: Tool chain is the bundle of compilers that contain multiple cross compilers to support different architecture of the target machine eg. 32 bit, 64bit etc. To compile the source code, hardware specific compiler is required. For example, to make a kernel for arm based processor, gcc cross compiler (ARM) is required for compilation. The link to download the source code and tool chain are https://github.com/raspberrypi/linux (Source Code), https://github.com/raspberrypi/tools.(Tool Chain)

Step 2 Configuring the source code

The kernel configuration and build system is based on multiple Makefiles. The main Makefile, present at the top directory of the kernel source tree which interact to all files of the source tree. Interaction takes place using the make tool, which parses the Makefile through various targets, defining what action should be done (configuration, compilation, installation, etc.). Run the command $**make help** at terminal to see all available configuration option. As shown in snapshot we created a folder "**cl_rpi**" and placed kernel source code, tool chain in order to shrink the path.

```
      @ms:~/cl_rpi/krnl-rpi$ make help
Cleaning targets:
  clean           - Remove most generated files but keep the config and
                    enough build support to build external modules
  mrproper        - Remove all generated files + config + various backup files
  distclean       - mrproper + remove editor backup and patch files

Configuration targets:
  config          - Update current config utilising a line-oriented program
  nconfig         - Update current config utilising a ncurses menu based program
  menuconfig      - Update current config utilising a menu based program
  xconfig         - Update current config utilising a QT based front-end
  gconfig         - Update current config utilising a GTK based front-end
  oldconfig       - Update current config utilising a provided .config as base
  localmodconfig  - Update current config disabling modules not loaded
  localyesconfig  - Update current config converting local mods to core
  silentoldconfig - Same as oldconfig, but quietly, additionally update deps
  defconfig       - New config with default from ARCH supplied defconfig
```

The kernel contains thousands of device drivers, file system drivers, network protocols and other configurable items. Thousands of options are available to selectively compile parts of the kernel source code. Configuration is the process of defining the set of options with which the target kernel to be compiled. The set of options depends upon:

- On your hardware (for device drivers, etc.)
- On the capabilities you would be include in to the kernel (network capabilities, file systems, real-time, etc.)

Kernel configuration is a step where the compiler is set to required configuration at the top level of source directory in .config file. This file can't be edited.

Kernel option types: - Kernel option type describes the state of hardware operation as given in the table.

Option Type	Configuration	Description
Bool	CONFIG_SYSVIPC=y	to include the feature in the kernel
	CONFIG_MMC=n	to exclude the feature from the kernel
Tristate	CONFIG_USB_USBNET=y	to include the feature in the kernel image
	CONFIG_USB_NET_PLUSB=m	to include the feature as a kernel module
	CONFIG_USB_NET_GL620A=n	to exclude the feature from the kernel
Int	CONFIG_MMC_BLOCK_MINORS=32	to specify integer values
String	CONFIG_LOCALVERSION="-quick"	to specify hexadecimal values
Hex	CONFIG_ZBOOT_ROM_BSS=0x0	to specify string values

Table 3.2 Kernel options

Step 3 Kernel Compilation and Building the Kernel Modules: -

Before compilation and building the kernel, the programmer or developer needs to understand the following:

Makefile: It is a text file written a defined syntax along with the Make Utility. It helps in building software from the source files. It is a way to organize the code, compilation and linking. The make utility is given as under:
$make
$make install
The above command installs the kernel modules at default lib folder of kernel source directory. Installs all modules in /lib/modules/<version>/kernel/

CROSS_COMPILE: -To compile a Linux kernel for different CPU architecture, cross compiler tool chain is required that compiles the kernel source code for targeted platform. For examples: mips-linux-gcc is the cross compiler meant for mips architecture similarly arm-linux-gnueabi-gcc, are the cross compiler for arm architecture. The CPU architecture and cross-compiler prefix are defined through the **ARCH** and **CROSS_COMPILE** variables in the top level Makefile. Where **ARCH** is the variable that has the name of the architecture for which the object code has to run. For example, ARCH= **arm** describes the cross compilation of a kernel for the arm architecture and **CROSS_COMPILE** is the prefix of the cross compilation tools chain. Example: arm-linux- if your compiler is arm-linux-gcc. The final command to cross compile the linux kernel for arm architecture is as:(Note Before issue command the user must be at the source directory of kernel in this case cl**_rpi/krnl-rpi** is kernel source directory)

$ make ARCH=arm bcmrpi_defconfig (For Raspberry Pi)
$ make ARCH=arm bcm2709_defconfig (For Raspberry Pi2) The output of the command given below

```
@ms:~$ cd cl_rpi/
@ms:~/cl_rpi$ cd krnl-rpi/
@ms:~/cl_rpi/krnl-rpi$ make ARCH=arm bcm2709_defconfig
#
# configuration written to .config
#
@ms:~/cl_rpi/krnl-rpi$
```

As a result of successful compilation a **.config** file is created at kernel source directory. This is the most common way of configuring a kernel for embedded and mobile platforms. A default configuration files is available, per board or per CPU family which are stored in **arch/<arch>/configs/**, where <arch> defines arm or any architecture. After successful configuration written to the kernel source directory, the following command is issued to compile the kernel. This command takes several minuts aprox 30-45 minuts to complete.

$ make -j<option> ARCH=arm CROSS COMPILE=/path of tool chain compiler/arm-linux-gnueabi-

Snapshot of this command given below

```
      @ms:~$ cd cl_rpi/
      @ms:~/cl_rpi$ cd krnl-rpi/
      @ms:~/cl_rpi/krnl-rpi$ make ARCH=arm bcm2709_defconfig
#
# configuration written to .config
#
      @ms:~/cl_rpi/krnl-rpi$ make ARCH=arm CROSS_COMPILE=/home/sanjay/cl_rpi/rpi-gcc/bin/arm-linux-gnueabihf-
scripts/kconfig/conf --silentoldconfig Kconfig
  CHK     include/config/kernel.release
  CHK     include/generated/uapi/linux/version.h
  CHK     include/generated/utsrelease.h
make[1]: `include/generated/mach-types.h' is up to date.
  CC      kernel/bounds.s
  GEN     include/generated/bounds.h
  CC      arch/arm/kernel/asm-offsets.s
```

As a result of successful compilation of kernel it's zImage is created at location "arch/boot/"

Step 4 Creation of img file

The created zImage at location "arch/boot/" is now converted into kernel.img file by using following command:

$/tool-master/mkimage/./imagetool-uncompressed.py /arch/arm/boot/zImage (For pi)

$/tool-master/mkimage/./mklimg –dtok zImage kernel7.img (For pi2) output of this command shown below and it creats the kernel7. img

```
@ms:~$ cd cl_rpi/
@ms:~/cl_rpi$ cd tools-master/
@ms:~/cl_rpi/tools-master$ cd mkimage/
@ms:~/cl_rpi/tools-master/mkimage$ ./mkknlimg --dtok zImage kernel7.img
```

Step 5 Creating modules for new kernel(kernel7.img)

Several utility programs of linux OS has to be integrated with the created new kernel image to form the complete Linux OS the command given below is executed and its output is given in screen shot.

$ make modules_install INSTALL_MOD_PATH=<dir>/(In the above command this)

```
  CHK     include/config/kernel.release
  CHK     include/generated/uapi/linux/version.h
  CHK     include/generated/utsrelease.h
make[1]: `include/generated/mach-types.h' is up to date.
  CALL    scripts/checksyscalls.sh
  CHK     include/generated/compile.h
  CHK     kernel/config_data.h
  Kernel: arch/arm/boot/Image is ready
  Building modules, stage 2.
  Kernel: arch/arm/boot/zImage is ready
  MODPOST 1369 modules
rammallik@rammallik:~/cl_rpi/kernal_rpi2$ make ARCH=arm CROSS_COMPILE=/home/rammallik/cl_rpi/tools-master/arm-bcm
2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin/arm-linux-gnueabihf- modules_install INSTALL_MOD_PATH=/home/
rammallik/Documents/
```

Note: - **INSTALL_MOD_PATH** is a variable used to specify the path of the modules to be installed. In this case the path is /home/rammalik/documents

Step 7 Combining the kernel.img and modules:

1. **Taking existing OS from web and write on the card.**
2. **The card having two partition boot and file system**
3. **Replace the existing kernel from boot partition**
4. **Replace the firmware and modules folder of file system partition of your module and firmware folder**

To transfer the built kernel, now the next step is to move the resulting kernel.img to the Raspberry Pi's **/boot/** directory. Two directories named firmware and modules are to be replaced from storage's (memory card) firmware and modules. The mentioned step is done in this manner:

Move the modules archive to the Raspberry Pi and extract them to firmware and module directories overwrite /lib/firmware and /lib/modules

The content of the mentioned directories is shown below as an example

./firmware

./firmware/brcm

./firmware/edgeport

./firmware/emi2

./modules

./modules/3.6.11+

./modules/3.6.11+/kernel

./modules/3.6.11+/kernel/lib

./modules/3.6.11+/kernel/fs

Now the final step is to reboot the Raspberry Pi to complete installation.

3.4 Customizing Android for Raspberry pi 2

Android is very popular operating system for various devices embedded environment. It is used on different types of embedded environment such as mobile phone, tablet, laptops etc. Android can easily portable on these devices and customizing it. Here we explain about one of these embedded board like Raspberry Pi 2 for android customization. Before we start it first we setup the environment of host machine (Linux, mac OS) to build an android for raspberry pi 2.

3.4.1 Downloading and Building

The Android build is on recent versions of Ubuntu LTS (14.04), but most distributions should have the required build tools available. Before you download and build the Android source, ensure your system meets the following requirements:

- A Linux or Mac OS system. It is also possible to build Android in a virtual machine on unsupported systems such as Windows. If you are running Linux in a virtual machine, you need at least 16GB of RAM/swap and 100GB or more of disk space in order to build the Android tree. See disk size requirements below.
- A 64-bit environment is required for Gingerbread (2.3.x) and newer versions, including the master branch. You can compile older versions on 32-bit systems.
- At least 100GB of free disk space for a checkout, 150GB for a single build, and 200GB or more for multiple builds. If you employ ccache, you will need even more space.
- Python 2.6 -- 2.7, which you can download from python.org.
- GNU Make 3.81 -- 3.82, which you can download from gnu.org,
- JDK 7 to build the master branch of Android in the Android Open Source Project (AOSP); JDK 6 to build Gingerbread through KitKat; JDK 5 for Cupcake through Froyo. See Initializing a Build Environment for installation instructions by operating system.
- Git 1.7 or newer. You can find it at git-scm.com.

3.4.2 Initializing a Build Environment

This section describes how to set up your local work environment to build the Android source files. You will need to use Linux or Mac OS. Building under Windows is not currently supported.

Some of the requirements for your build environment are determined by which version of the source code you plan to compile. You may also choose to download and build the latest source code (called master), in which case you will simply omit the branch specification when you initialize the repository.

Once you have selected a branch, follow the appropriate instructions below to set up your build environment.

3.4.3 Setting up a Linux build environment

These instructions apply to all branches, including master. The Android build is routinely tested in house on recent versions of Ubuntu LTS (14.04), but most distributions should have the required build tools available. Reports of successes or failures on other distributions are welcome.

Installing the JDK

The master branch of Android in the Android Open Source Project (AOSP) requires Java 7. On Ubuntu, use Open JDK.

Java 7: For the latest version of Android

$sudo apt-get update
$ sudo apt-get install openjdk-7-jdk

Installing required packages (Ubuntu 14.04)

You will need a 64-bit version of Ubuntu. Ubuntu 14.04 is recommended.

$ sudo apt-get install bison g++-multilib git gperf libxml2-utils make python-networkx zlib1g-dev:i386 zip

3.4.4 Downloading the Source

The Android source tree is located in a Git repository hosted by Google. This document describes how to download the source tree for a specific Android code-line.

Installing Repo

Repo is a tool that makes it easier to work with Git in the context of Android. For more information about Repo, see the Developing section.

To install Repo:

1. Make sure you have a bin/ directory in your home directory and that it is included in your path:

   $ mkdir ~/bin
   $ PATH=~/bin:$PATH

2. Download the Repo tool and ensure that it is executable:

   $ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
   $ chmod a+x ~/bin/repo

Initializing a Repo client

After installing Repo, set up your client to access the Android source repository:

1. Create an empty directory to hold your working files. If you're using Mac OS, this has to be on a case-sensitive file system. Give it any name you like:

$ mkdir WORKING_DIRECTORY
$ cd WORKING_DIRECTORY

2. Run repo init to bring down the latest version of Repo with all its most recent bug fixes. You must specify a URL for the manifest, which specifies where the various repositories included in the Android source will be placed within your working directory.

$ repo init -u https://android.googlesource.com/platform/manifest

To check out a branch other than "master", specify it with -b

$ repo init -u https://android.googlesource.com/platform/manifest -b android-5.1.1_r8

3. When prompted, configure Repo with your real name and email address. To use the Gerrit code-review tool, you will need an email address that is connected with a registered Google account. Make sure this is a live address at which you can receive messages. The name that you provide here will show up in attributions for your code submissions.

A successful initialization will end with a message stating that Repo is initialized in your working directory. Your client directory should now contain a .repo directory where files such as the manifest will be kept.

Setting up android for Raspberry pi2 we add local mainfests file in to the .repo directory by

$ cd .repo
$ git clone https://github.com/peyo-hd/local_manifests

3.4.5 Downloading the Android Source Tree

To pull down the Android source tree to your working directory from the repositories as specified in the local manifest, run

$ repo sync

The Android source files will be located in your working directory under their project names. The initial sync operation will take an hour or more to complete depending on your internet speed. After successful completion of download the android source code we start building the android for Raspberry pi2 by issue the command.

$ source build/envsetup.sh
$ lunch rpi2-eng
$ make

After completion of android making we build the kernel for raspberry pi2 and port them on it.

$ cd kernel/rpi
$ ARCH=arm scripts/kconfig/merge_config.sh arch/arm/configs/bcm2709_defconfig android/configs/android-base.cfg android/configs/android-recommended.cfg
$ ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- make zImage
$ ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- make dtbs

Preparing the SD Card:

Partitions of the card should be set-up like followings.

p1 512MB for BOOT : Do fdisk, format as fat32, tag bootable flag
p2 512MB for /system : Do fdisk, new primary partition, size 512M.
p3 512MB for /cache : Do fdisk, format as ext4
p4 remainings for /data : Do fdisk, format as ext4

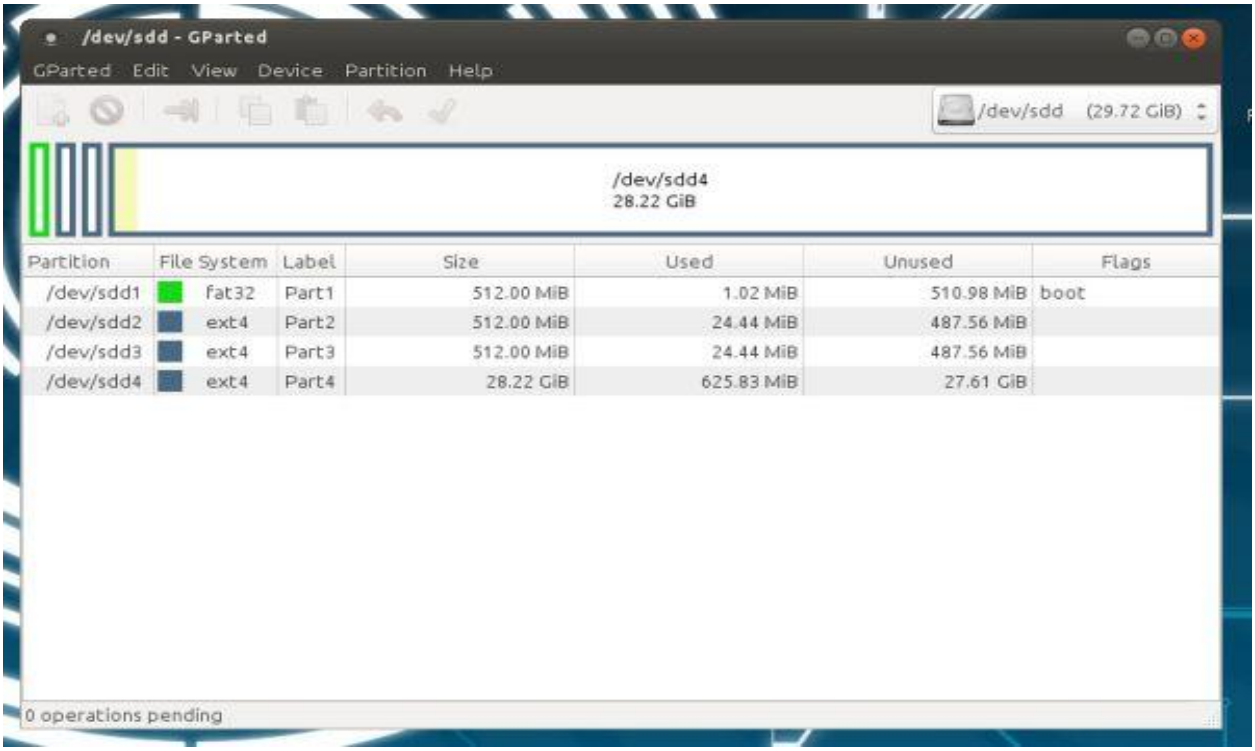


Fig.3.2 Memory Card Partition (Android)

Write system partition on SD Card Partition

$ cd out/target/product/rpi2
$ sudo dd if=system.img of=/dev/<p2> bs=1M

BootPartiton,Kernel and Ramdisk copy to SD Card Partition

device/brcm/rpi2/boot/* to p1:/

kernel/rpi/arch/arm/boot/zImage to p1:/

kernel/rpi/arch/arm/boot/dts/bcm2709-rpi-2-b.dtb to p1:/

out/target/product/rpi2/ramdisk.img to p1:/

3.4.6 Installing Android OS on Raspberry Pi

Android has vast applications and can be run on raspberry pi. This section explains how to download & install android OS image on Raspberry Pi.

Step 1: Download the image file of android OS from the link as (with name CynogenMOD 7.2) on Laptop/PC: **http://androidpi.wikia.com/wiki/Android_Pi_Wiki**

Step 2: Download Win32 software to burn/write the image file on the SD card from the link **http://sourceforge.net/projects/win32diskimager/**

Step 3: Extract the Image file.

Step 4: Insert SD Card into pc/laptop (system on which OS and win32disk is downloaded). Make sure that SD card is empty and have atleast 4GB memory.

Step 5: Open win32 disk. Its UI as shown in Fig-1 . Browse the image file which has been downloaded.

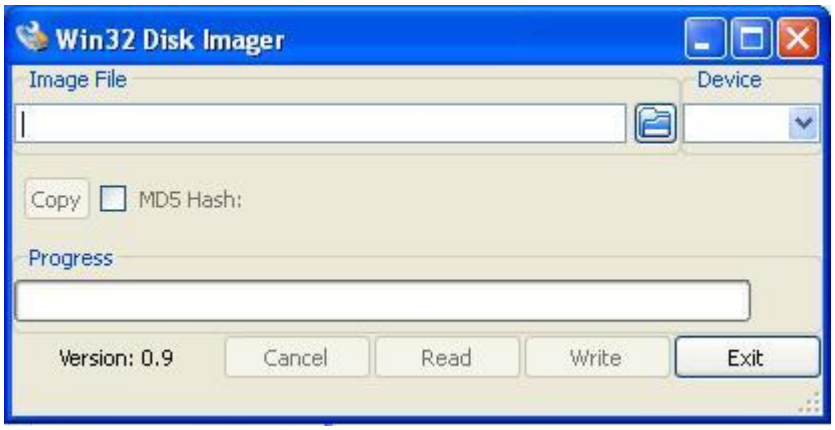


Fig 3.3. Win 32 disk imager.

Step 6: Click on "write". When writing is completed, remove the SD card and insert into the raspberry pi.

Step7: Power ON Raspberry Pi. After few minutes android screen will appear on the display

connected through HDMI port of raspberry Pi which is shown in the Fig-2.

Note- Since the processor speed is slow and memory is less any operation thereafter takes more time.

Fig 3.4. Android on Raspberry Pi.

Exercises

1. Study about different Linux versions that can be used as a mobile OS for Raspberry Pi.
2. Customize the kernel module for different requirements as per the applications.

References:

[1] http://windowsphone.interoperabilitybridges.com/articles/qt-to-wp7-chapter-1-introducing-windows-phone-platform-to-symbian-qt-application-developers

[2] http://thinkingweb.co/introduction-android-platform/

[3] http://www.gamedev.net/page/resources/_/technical/mobile-development/introducing-xcode-tools-for-iphone-development-r2877

Chapter- 4
Design and Development of Smart Phone

4.1 Introduction

A smart phone or mobile device is a handheld portable, compact and lightweight device with sufficient storage and processing power and small display. Technological advance has evolved these devices from basic mobile phones to future phones and today's smart phones. Today's high end smart phone has the capability similar to any basic desktop or laptop. A basic layered architecture of the mobile phone comprises of Hardware**,** Operating System (OS)**,** and Application Software. This chapter briefly explains the design methodology of a smart phone with the help of Raspberry Pi.

4.2 Design lifecycle

The design cycle/flow of a mobile device is as shown in the figure 4.1. Each phase of the design cycle is explained in following sections:

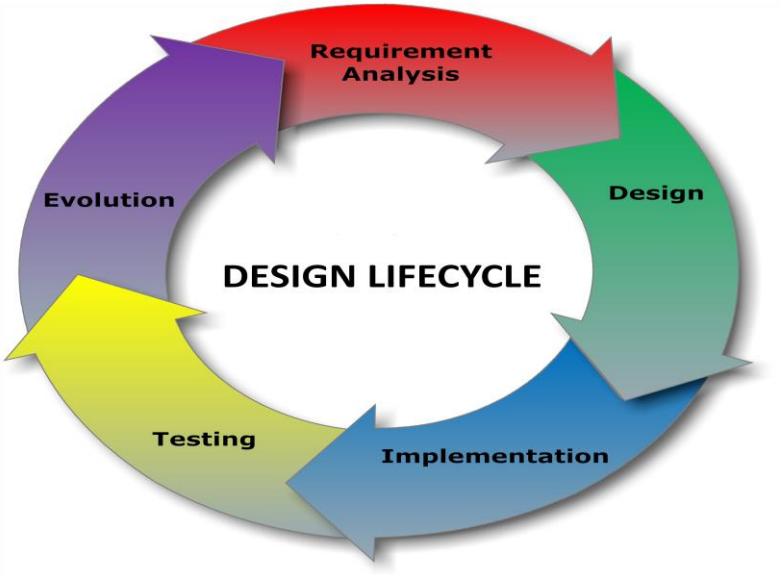


Fig 4.1: Design Lifecycle

1. **Requirements Analysis:** This phase consists of understanding the requirements of the project by interacting with the user. This is an informal way of noting down all the materials/requirements that are required by the user. In this book the Raspberry Pi board is used as the main processing unit in order to design the smart phone with the features of GPS, Bluetooth, GSM communications, TFT, MIC, Speakers and Camera for multimedia applications and various sensors for environmental and health related parameter sensing.
2. **Specifications**: This phase deals with the detailed description of all the components and its specifications in terms of size, height, width and quantity. The various components used in this book are Raspberry Pi camera, GSM modem, GPS module, Bluetooth module and various sensors. The specification of each device is given in appendix.

3. **Designing:** This phase consists of making the pin diagram, architectural diagram, block diagram of smart phone for visual representation. It consists both hardware and software design. The designing phase is discussed in the respective chapters.
4. **Implementation**: This phase consists of structure created in order to check the functionality of the product before the final development. The implementation phase is discussed in figure 1.5. Similar to designing phase, this phase is also discussed in the respective chapters.
5. **Testing:** Testing helps to check the functionality of the product to see if it is as per our requirement or not. The testing phase is discussed in the respective chapters.
6. **Deployment:** Finally the product is released and available for the public use. In this case smartphone is the final product with the feature of touch screen, camera, GSM,GPS, Bluetooth and few sensing modules.

The Design Approaches are describes in the following section:-

The design phases can be Top Down or Bottom Up which are explained in the following section. The Top Down approach is used to develop a new product which does not exist earlier and the design tasks are decomposed into compositional sub-systems. In a top-down approach an overview of the system is formulated from requirements phase to testing phase as explained in the previous section.

Bottom-up approach is used to modify the features of the existing system or to add new features in to the existing system. It follows the bottom to top approach from implementation to meeting the requirements in reverse order. This approach is not suitable for a completely new system design. In a bottom-up approach the individual base elements of the system are first specified in great detail. These elements are then linked together to form larger subsystems, which then in turn are linked, sometimes upto many levels, until a complete top-level system is formed. This strategy often resembles a "seed" model, whereby the beginnings are small but eventually grow in complexity and completeness.

4.3 Component Selection

In order to design, selection of components to meet the requirements is important. The selection of components depends on device under development. The major components needs to be selected are:

1. System on Chip(SoC)
2. Processor
3. Input/ Output and other Peripherals
4. Communication Module
5. Operating System
6. Programming language
7. Other factors includes cost, tool support, memory and availability and performance etc

SoC: The Broadcom SoC used in the Raspberry Pi 2 has the capability to be used as a SoC of a mobile device. While operating at 900 MHz by default, the Raspberry Pi 2 provides a real world performance. This SoC doesn't require heat sink or special cooling mechanism. The graphics capabilities of the Raspberry Pi are roughly equivalent to the level of performance of the Xbox of 2001. The SoC is stacked underneath the 1 GB of RAM.

Processor: Raspberry Pi uses the ARM11 micro-architecture that supports SIMD media instructions, multiprocessor support and new cache architecture. The implementation included a significantly improved instruction processing pipeline, compared to previous ARM9 or ARM10 families. The ARM11 family is currently the only ARMv6-architecture core.

Input/ Output and other peripherals: A peripheral is a device that is used to put information into or get information out of the computer. There are three different types of peripherals:

1. Input, used to interact with, or send data to the mobile device (keypad, mic, touch screen etc.)
2. Output, which provides output to the user from the mobile device (touch screen, speaker, etc.)
3. Storage, which stores data processed by the mobile device (RAM, flash drives, memory card etc.)

Communication Module:

1. Wireless: Bluetooth, Wi-Fi, GSM, GPS
2. Wired: UART, SPI, I2C

Operating System: Raspberry Pi supports various Linux distributions such as Raspbian, Fedora etc.

Programming Language: Python, C, Perl and Ruby. Python is the main programming language.

4.4 Introduction to Raspberry Pi:

Raspberry Pi is a credit card sized small CPU or a complete computing platform developed by Raspberry Pi foundation in UK and released in February, 2012. It is released to enhance the school level education. It supports different distributions of linux but recommended is **Raspbian**. Raspbian is a new operating system with the use of Debian by Raspberry Pi foundation i.e. Raspberry + Debian = Raspbian.

The Raspberry Pi foundation provides OS like Raspbian,Noobs,Snappy Ubuntu Core,Openelec,Osmc,pidora,risc OS etc. on their official website **www.raspberrypi.org**. The most important features of Raspberry Pi is its processor's speed and its small size. It is very much small according to its powerful features.
Raspberry Pi model A+ was released followed by B, B+, Raspberry Pi 2. There are some basic changes in hardware with the changes in the board version.

Block Diagram of Raspberry Pi

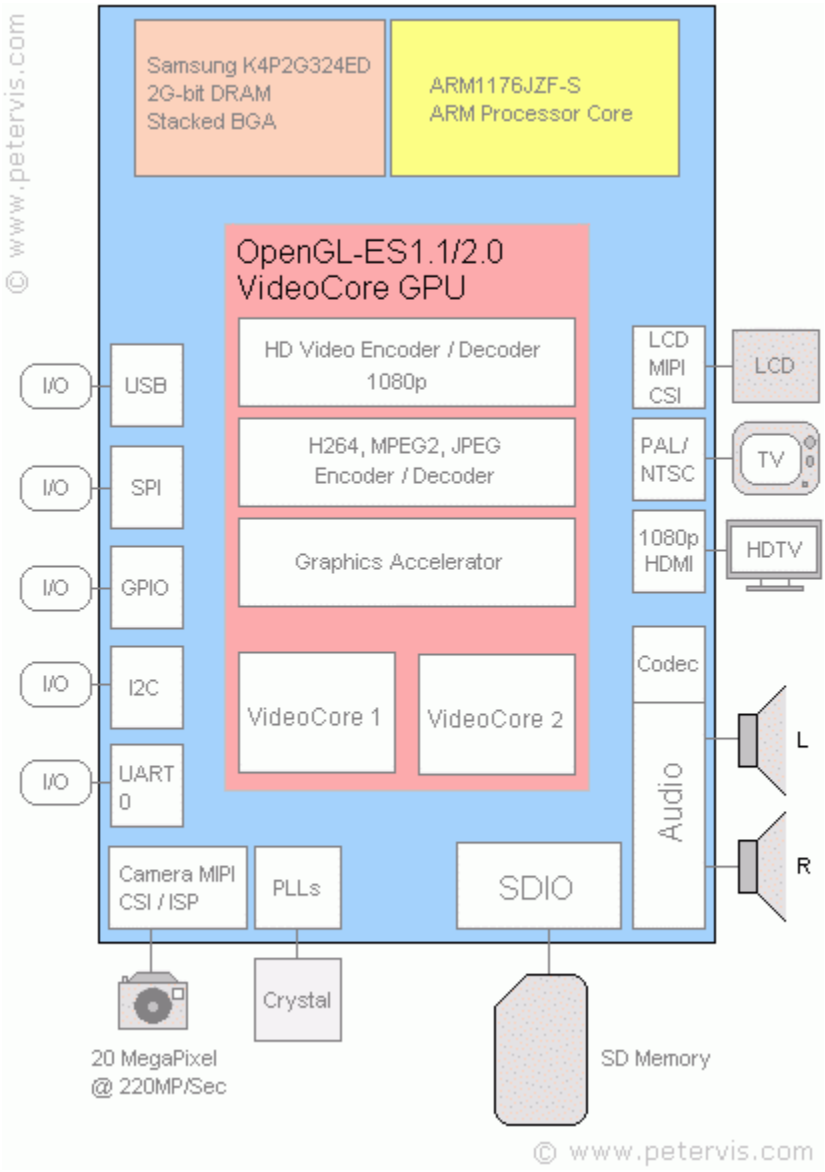


Fig 4.2 Block diagram of an SOC [1]

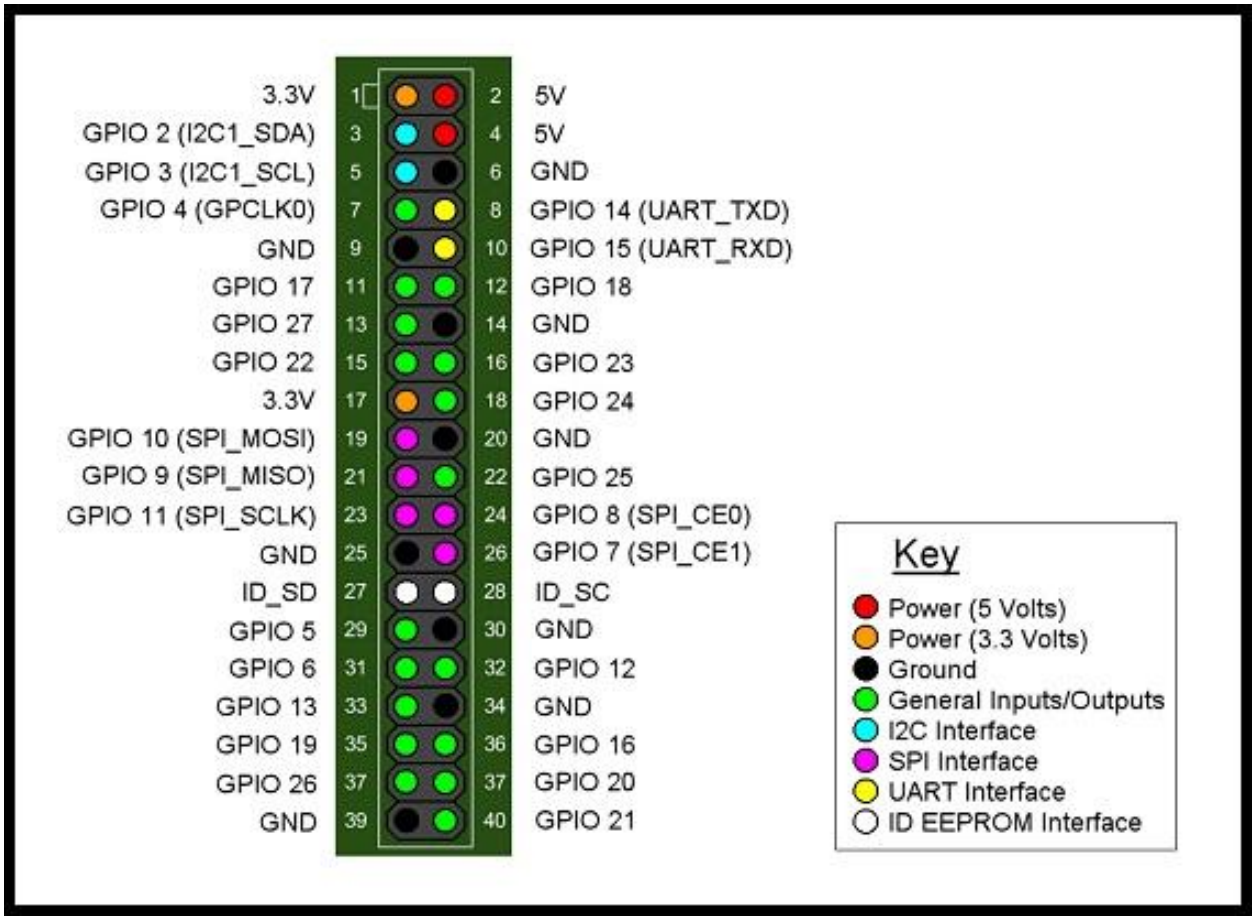


Fig 4.3 Pin Diagram [2]

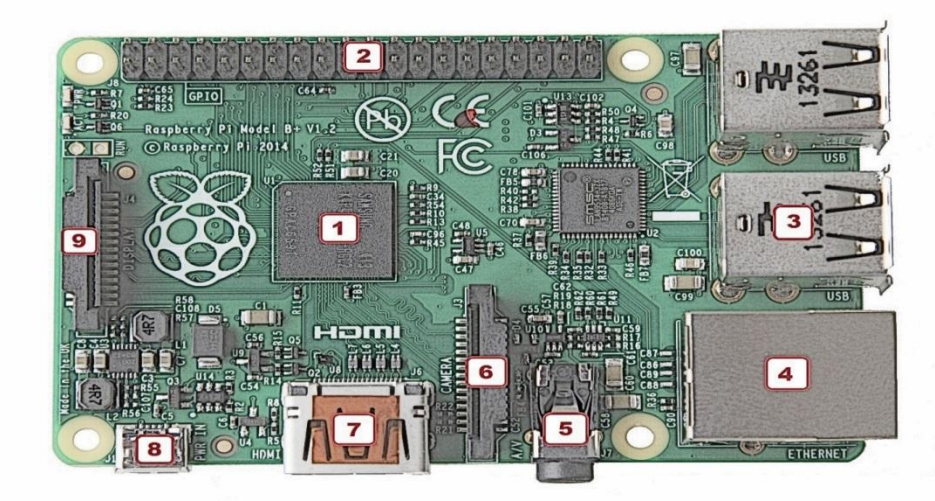


Fig: 4.4 Raspberry Pi Front side

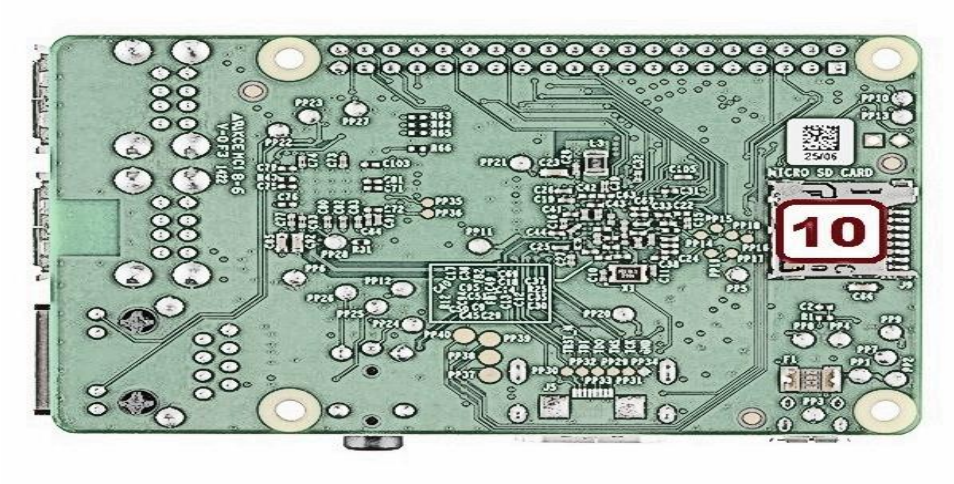


Fig: 4.5 Raspberry Pi Back side

Description:

1. **Processor**: Raspberry Pi has ARM11 processor which is a 32 bit processor and having speed 900MHz.
   It has Broadcom SOC (System on Chip) that is BCM2836.
2. **Input/ Output Pins**: There are 40 input/output pin in model 2 and B+. The previous models have 26 input/output pins. The expansion header is 2.54 mm (2x20 strip).There are 27 GPIO (General purpose Input Output) with +3.3V, +5V and GND pins.

3. **USB Ports**: Model Pi 2 and B+ have four 2.0mm connector USB ports. The previous models have only two USB ports. The Model B+ eliminates the use of a USB Hub.

4. **Ethernet Port**: Model B and B+ have 10/100 Base T Ethernet socket but previous version do not have this capability.

5. **Audio Jack**: Raspberry Pi has 3.5mm audio jack.

6. **Camera Connector**: Raspberry Pi has MIPI 15-pin camera serial interface (CSI-2)

7. **HDMI port:**All the Raspberry Pi model have HDMI(rev 1.3 & 1.4) composite RCA(PAL & NTSC)

8. **Power:** It's a micro USB socket (5V, 2A).

9. **Display Connector**: It's a Display Serial Interface (DSI) 15 way flat flex cable connector with two data line and a clock lane.

10. **SD Card Slot:** It's a micro SD card slot. The minimum capacity of a card should be 4 GB.

Differences between Raspberry Pi 2 and Raspberry B+ :

	Raspberry Pi 2	**Raspberry Pi B**
Processor Chip Set	Broadcom BCM2836	Broadcom BCM2835
RAM	1 GB	512 MB
Storage	Micro SD	Micro SD
USB 2.0	4*USB Ports	4*USB Ports
CPU	900 MHz quad-core ARM Cortex-A7	700 MHz single-core ARM1176JZF-S
GPIO	40	26
Ethernet Port	YES	YES

4.5 Running Raspberry Pi Headless

Raspberry Pi needs to be plugged with monitor, keyboard and mouse. Running Raspberry Pi without these is called as Running Raspberry Pi Headless.

Headless running of Raspberry Pi can be achieved in two manners, using onboard UART or over a network.

4.5.1 Over a network through SSH Secure Shell

Command line of a Raspberry Pi can be used from another computer on the same network using SSH. SSH is a cryptographic network protocol for secure data communication, remote command-line login, remote command execution, and other secure network services between two networked computers. It connects, via a secure channel over an insecure network, a server and a client running SSH server and SSH client programs, respectively.

Procedure

1. Power on Raspberry Pi.
2. On windows PC, download SSH client. The most commonly used SSH client is PuTTY.
3. In PuTTY, one can create profiles for connections to various SSH servers.
4. Connect ethernet cable with known IP address on a local network to Raspberry Pi.
5. Start PuTTY by double-clicking its executable file.
6. In the category Session (see the tree on the left side of fig4.6), enter IP address of Raspberry Pi under Host Name, enter 22 under Port and select SSH under Protocol.
7. Then go to Connection -> Data and specify the username to log in.
8. Under SSH**, enable X11.**
9. Connect to SSH server by clicking on Open.
10. If nothing happens for a while after clicking ”**OPEN**” button or message prompts as ”**Network Error: Connection Timed Out**”, it’s likely that wrong IP address for Pi is entered.

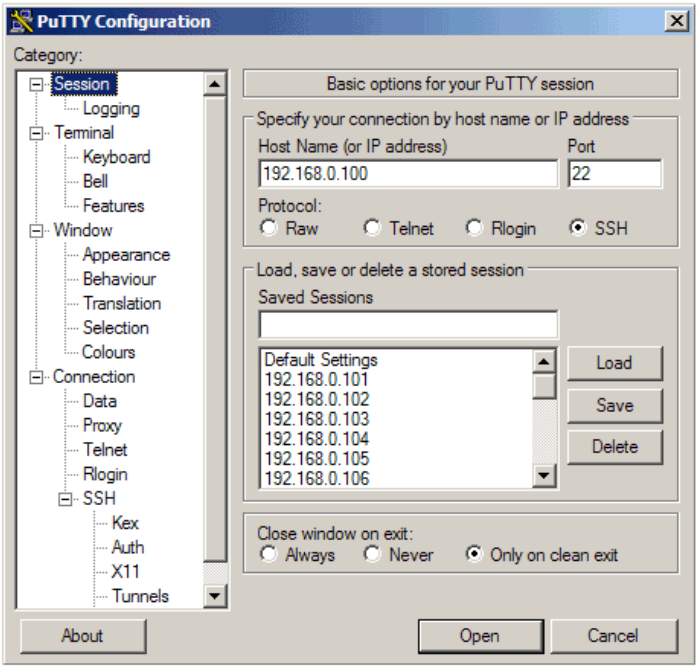


Fig:4.6 PuTTY Configuration

11. When connection works, ignore security warning below and click “YES” button.

Fig:4.7 Putty Configuration

12. Usual login prompt will appear on connection, login with the username & password as **pi** and **raspberry** respectively
13. Start Xming on PC.
14. In the putty window, type “**startlxde**” to open the LXDE desktop.

15. Graphical view of Raspberry Pi desktop will be available on PC.

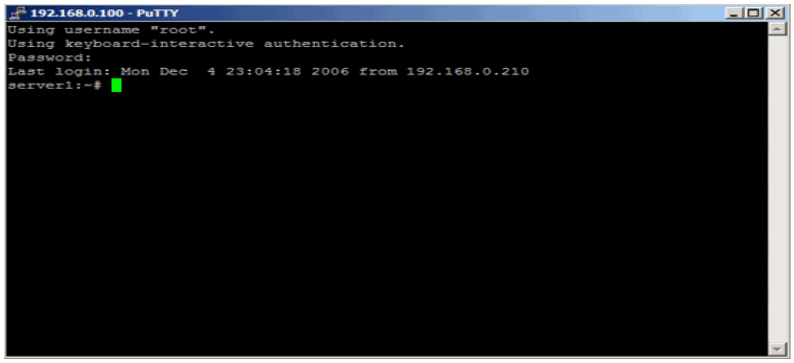


Fig: 4.8 PuTTY Terminal

4.5.2 Running Raspberry Pi Headless: Using onboard UART interface with Terminal application

Display messages while booting, are available by default at, on board UART pins (on GPIO14, GPIO15) of Raspberry Pi's and may be used to access Raspberry Pi on command line. It is an easy and cheap way to use Raspberry Pi but with disadvantage of non-availability of graphical view and restriction of that UART usage.

A number of USB to UART converter board are available e.g. FT232RL based, CP2102 based, PL2303 based. USB to UART convertor will be required for communication between regular desktop/laptop and Raspberry Pi. It is a breakout-module to convert USB to UART and vice versa. UART of this module is directly connected to Raspberry Pi's Rx-Tx lines and USB is directly connected to USB port of PC. Module source the power from the USB Bus, i.e. from PC / Laptop.

A number of terminal applications are available for viewing data communication like Terminal, Real Term etc

Block Diagram

Fig: 4.9 Block Diagram of Interconnection

Procedure

1. Download terminal application on PC. Here Real Term is used.
2. Download applicable driver for USB to UART converter interface.
3. Plug USB cable at PC
4. Open the device manager to check, to which COM port is assigned to the converter board.
5. Cross connect UART pins (TXD, RXD) with Raspberry Pi UART as shown in connection diagram (Fig. 4.10)
6. In the Real term app set following parameters:
    i. Baud rate: 115200
    ii. Hardware flow control: None
    iii. COM Port
    iv. Port: VCP to which the interface board has been established
    v. Parity: None, Start bit: 8, Stop bit: 1.
    vi. Press Open tab
7. Power on Raspberry Pi
8. Boot messages will be displayed on real term terminal window (Fig 4.11).

Connection Diagram

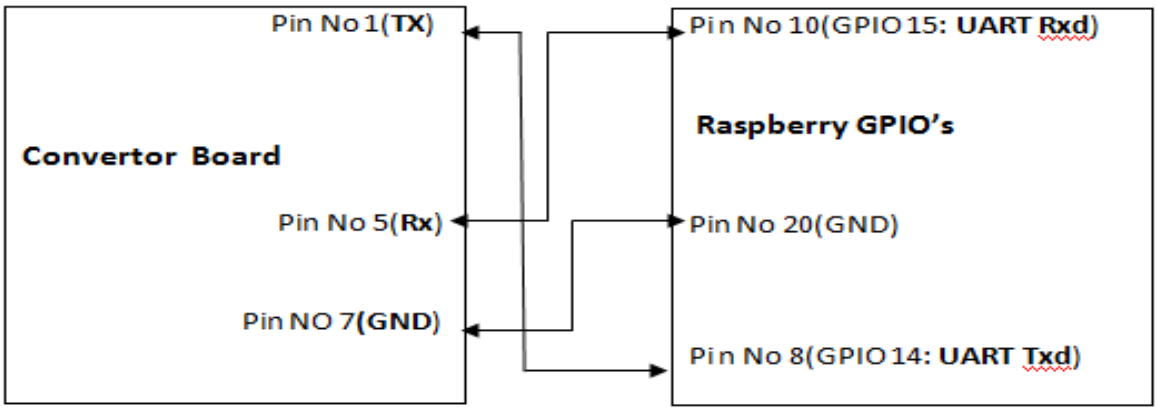


Fig4.10: Connection Diagram

Output

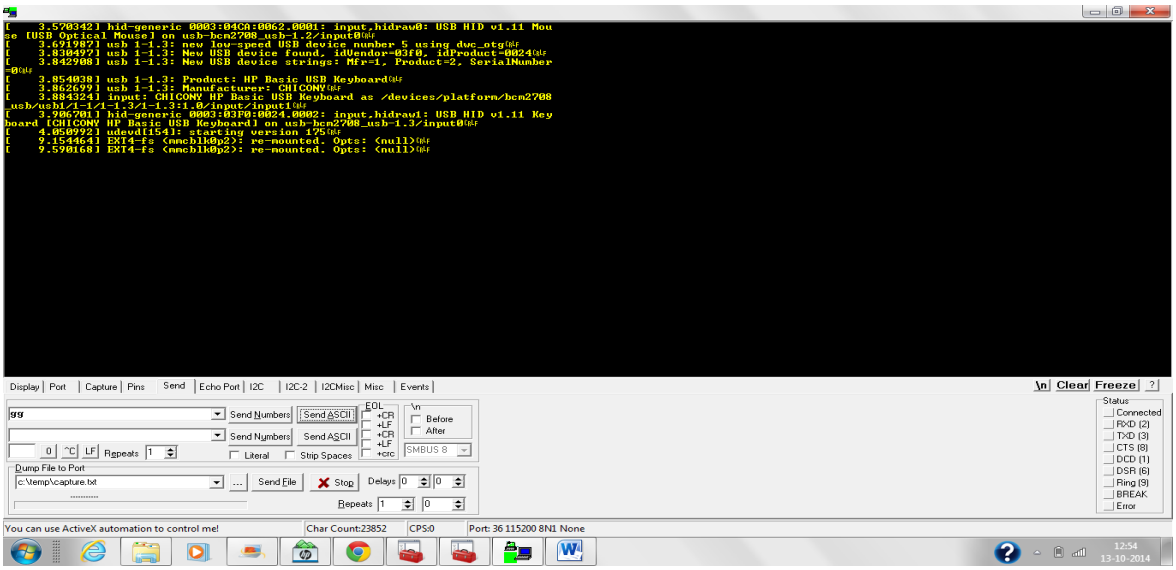


Fig4.11: Boot messages at Real Term window

4.5.3 Running Raspberry Pi Headless: Using onboard UART interface with PuTTY

1. Open the PuTTY terminal and enable “Serial” option
2. Enter the COM line to which the line has been established by USB to UART interface.
3. Under SSH-Serial set the serial parameters as
    i. Speed: 115200
    ii. Flow control: None
    iii. Parity: None, Start bit: 8, Stop bit: 1.
4. Press Open tab
5. Boot messages will be displayed on putty window

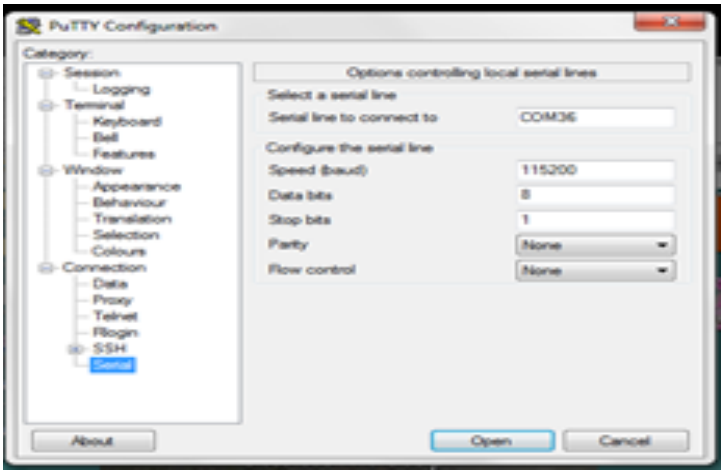


Fig4.12: PuTTY settings for serial interface

References:

1. http://www.petervis.com/Raspberry_PI/BCM2835_Block_Diagram/BCM2835_Block_Diagram.html.
2. http://blog.oscarliang.net/use-gpio-pins-on-raspberry-pi/.
3. http://www.penguintutor.com/linux/raspberrypi-headless
4. https://www.raspberrypi.org/forums/viewtopic.php?f=91&t=74176
5. http://blog.self.li/post/63281257339/raspberry-pi-part-1-basic-setup-without-cables
6. http://raspberrypi.stackexchange.com/questions/15192/installing-raspbian-from-noobs-without-display

Chapter 5
Display Interfacing and Programming

5.1 Introduction

Display forms an integral part of smart device to communicate with the user. Displays like LED, Seven Segment, LCD, Touch screens or TFT are widely used in smart devices to display status of device. Display interfacing and programming is described in this chapter.

Raspberry Pi2 board supports display output for composite video (NTSE and PAL) via an RCA plug, HDMI 1.3a standard output, Display Serial Interface (DSI)– via unpopulated 15 way flat flex connector. Raspberry Pi HDMI connector can be directly connected to an HDMI monitor using a standard HDMI cable. DSI connector is on board on the Raspberry Pi. Graphical Liquid Crystal Display (GLCD)/ Organic Light-Emitting Diode (OLED) displays, LEDs and Seven-Segment displays can also be interfaced via GPIO. In this chapter most of these display units are interfaced with Raspberry Pi module using I2C or DSI or GPIO.

5.2 Types of Displays

5.2.1. Interfacing LED

LED can be interfaced to any GPIO of Raspberry Pi as per the interfacing diagram shown in figure 5.1. LED's anode is connected at pin no 22 i.e.GPIO25 through 330 ohms resistor and cathode at ground.

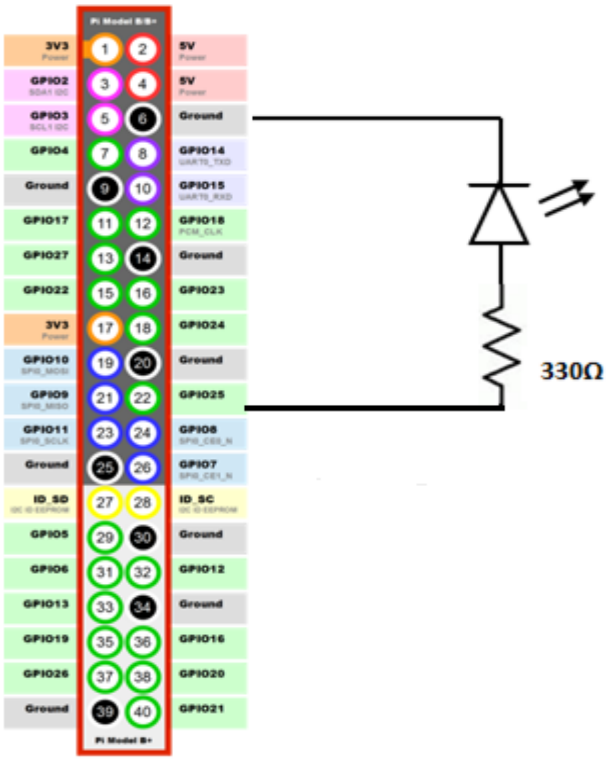


Figure 5.1 interfacing diagram for LED

After proper interfacing, open LeafPad editor and write python code for toggling LED with 2 second delay.

5.2.1.1 Python Code

```
import RPi.GPIO as GPIO                 # Import GPIO library
import time                             # Import time library
GPIO.setmode(GPIO.BCM)                  # Set GPIOs of Raspberry pi for BCM2836 architecture
GPIO.setup(25, GPIO.OUT)                # Configure GPIO 25 as an output pin
while True:                             # Continuous loop for togging
        GPIO.output(25, GPIO.HIGH)      #set GPIO 25 High(LED On)
        time.sleep(2)                   #set time delay of 2 sec for LED toggling
        GPIO.output(25, GPIO.LOW)       #set GPIO 25 Low(LED Off)
        time.sleep(2)                   #set time delay of 2 sec for LED toggling
```

5.2.2 Seven Segment Display

Seven Segment displays can be considered as combination of 7 LEDs into one module and are used mainly for numeric displays with little character information like real time clocks, event counters etc. They are available in two type namely common cathode and common anode as shown in figure 5.2

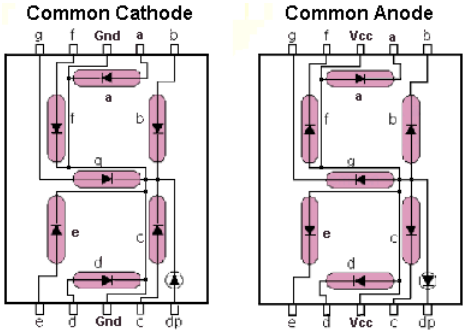


Fig 5.2 Seven segment displays [5]

5.2.2.1 Interfacing of Seven Segment Display with Raspberry Pi2

Commonly used method for interfacing seven segment displays with any controller is by multiplexing to save I/O pins. Here all segments are connected at same I/O pins but are derived differently using individual I/O pins.

Fig 5.3 shows interfacing diagram of a seven segment displays with Raspberry pi 2.

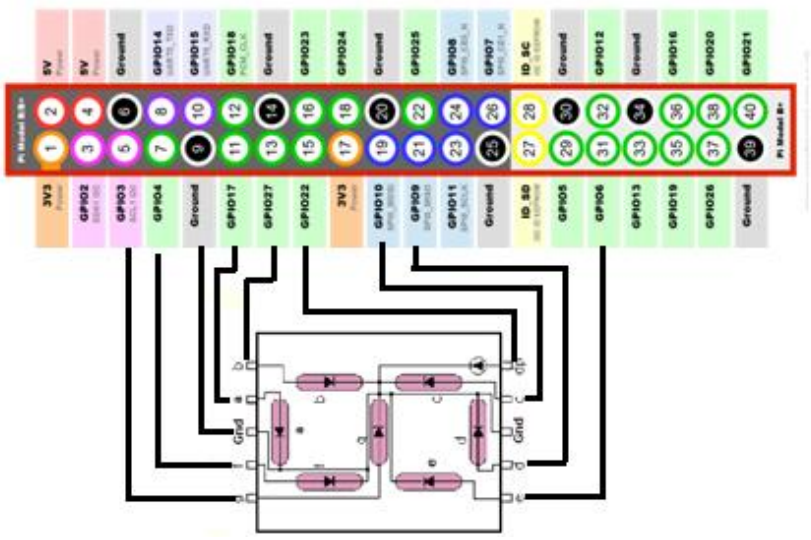


Fig 5.3 Interfacing diagram for Seven Segment Displays

5.2.3 Character cell LCD module

Character cell LCD module have inbuilt controllers and interfaces for feeding in text and symbols either for alphanumeric type or for graphical type with screen sizes 16x2 to 40x4. These can be commonly controlled over UART or through GPIOs to display values.

5.2.3.1 Necessary installations: To run the LCD using python libraray

$sudo apt-get install git

$ git clone git://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git

This will download Adafruit-Raspberry-Pi-Python-Code folder from github. This folder contains a file Adafruit_CharLCD.py that contains necessary code for LCD interfacing. Copy this file to your current directory. A slight change is needed in this file. Open file in text editor and search for line

def __init__(self, pin_rs=25, pin_e=24, pins_db=[23, 17, 21, 22], GPIO = None): Change this with

def __init__(self, pin_rs=25, pin_e=24, pins_db=[23, 17, 27, 22], GPIO = None): And save.

Connect 16x2 Alphanumeric LCD with raspberry pi as per the interfacing diagram as shown in figure 5.4. Control lines RW connected to GND and RS and Enable connected to Raspberry PI GPIO and LCD is interfaced with Raspberry Pi in 4-bit mode at data lines D4-D7.

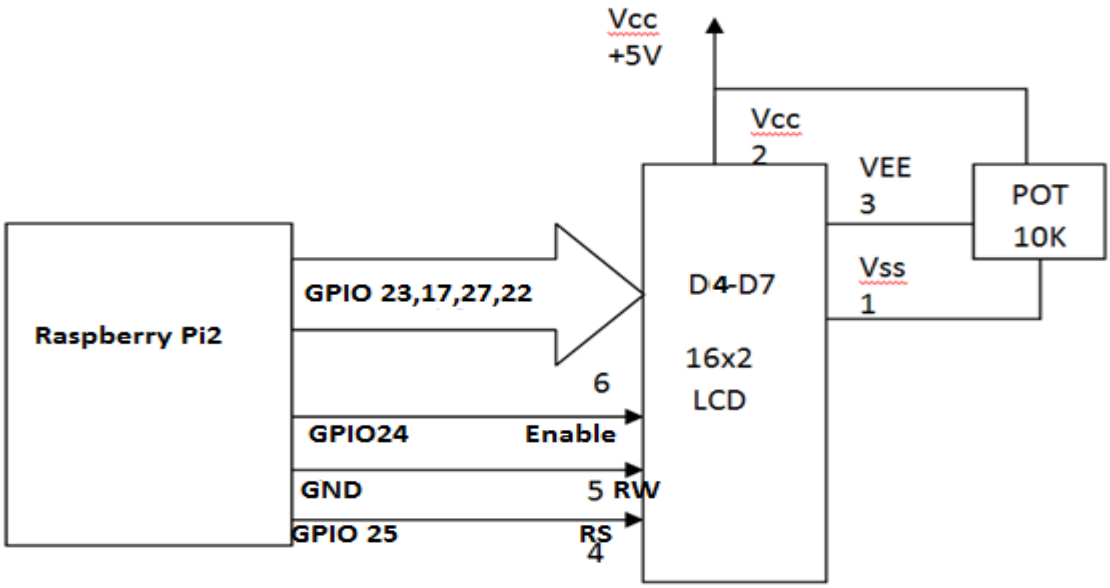


Fig 5.4 Interfacing diagram for 16x2 LCD

5.2.3.2 Python code

```
#Message " IGDTUW  welcomes you "  on 2 lines of 16x2 Alphanumeric LCD
    from Adafruit_CharLCD import Adafruit_CharLCD     #Import LCD control library file
    from time import sleep                                    #Import time library file
    lcd = Adafruit_CharLCD()                                      # Create lcd object as CharLCD
    type
    lcd.begin(16,2)                                           # Choose 16x2 size LCD
    while 1:
        lcd.clear()                                           # Clear LCD first
        lcd.message('          IGDTUW        \n      Welcomes you      ')    # Welcome
    message on LCD
        sleep(1)                                          # Wait for 1 second
```

5.2.3.3 LCD Output from Raspberry pi

Fig 5.5 Output at 16x2 LCD

5.2.4 Thin Film Transistor-Liquid Crystal Display (TFT LCD) Module

I2C/SPI is used to interface 4 pin resistive Touch Screens for serving both keyboard and display purposes. These are available in sizes from 1.8" to 2.8". Parallel interface displays are available in many sizes, usually up to 7" and more. These are usually 8 or 16-bits or 18 or 24-bit wide with some control-lines. Raspberry Pi does not contain enough GPIOs for 16-bit wide parallel displays but this could be solved by borrowing some GPIOs from the CSI-connector or from P5. Alternatively, some additional electronics (e.g. shift-registers or a CPLD) can be used which could also improve the frame rate or lower the CPU-load.

5.2.4.1 TFT Interfacing with Raspberry Pi

Interfacing of a serial TFT LCD using SPI serial interface is described in this section. It uses the hardware SPI pins (SCK, MOSI, MISO, CE0, CE1) as well as GPIO No 24 and 25. SPI is a bidirectional protocol with two separate data lines namely MOSI and MISO respectively. MOSI (Master-out, Slave-in) is needed to send the data from the Raspberry Pi to the display. This TFT provides a 3.5” display for the Raspberry Pi featuring a 3.5” TFT LCD 320x480 resolution display with touch screen. This TFT LCD connects directly to the GPIO connector of the Raspberry Pi. The interfacing between TFT and Raspberry Pi will be done as per the table given below.

Raspberry Pi2 Pin	Raspberry Pi2 GPIO	TFT Connections
1	3.3V	3.3V
18	GPIO24	RS (Register Select)
19	GPIO10/MOSI	MOSI/DIN (Digital Input)
24	GPIO8/CE0	CS (Chip Select)
23	GPIO11/CLK	SCLK/DCLK (Serial/Digital clock)
22	GPIO25	RST (Reset)
6	GND	GND (Ground)
21	GPIO9/MISO	DOUT (Digital Out)
29	GPIO5	Touch Screen PENIRQ
26	GPIO7/CE1	Touch Screen CS (Chip select)
12	GPIO18	Touch Screen BL (Backlight)

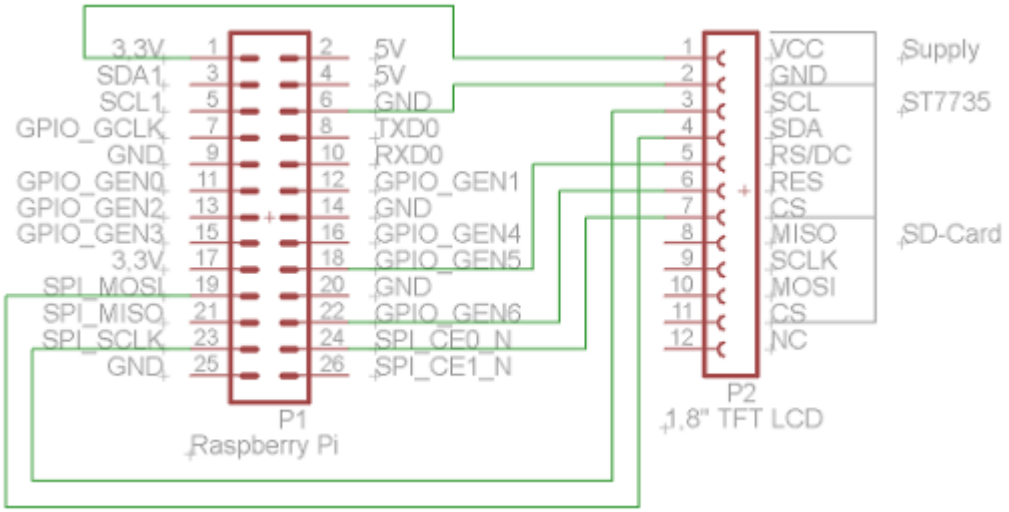


Fig 5.6 Example of TFT interfacing diagram [7]

TFT that is interfaced is 3.5" (320x480 pixels) Color TFT SPI LCD Module with touch screen with PCB for Raspberry Pi 2. This TFT LCD connects directly to the GPIO connector of the RPi comes in 83 mm X 63 mm X 20mm (with breakout board) dimensions. Resistive touch- screen and touch screen controller is added with this TFT. Additional GPIO connector on TFT PCB is given to use rest of the GPIO pins for other use and is powered directly from the RaspberryPi2[8].

Connect TFT with Raspberry Pi board as shown in figure 5.6 at GPIO and follow these steps.

1. Open "TinyLCD35_..._image" Folder from TFT DVD. Unzip image file.
2. Write this image to SD card using disk image writer. As shown in figure 5.7

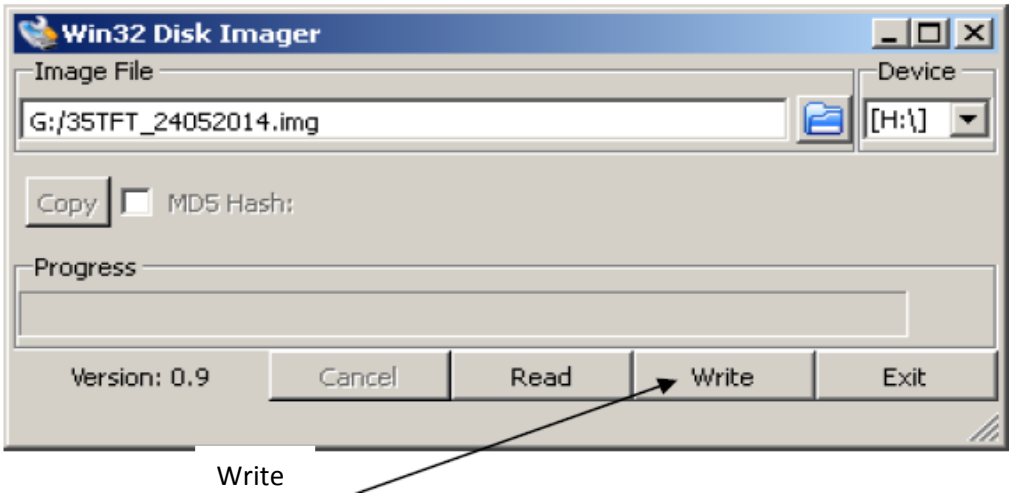


Fig 5.7: Image writing process on SD Card.

3. Login with this card with username **pi** and password **raspberry. The output of the OS on TFT screen is shown in the figure 5.8**

Fig 5.8 Outputs and TFT interfaced with Raspberry Pi

5.2.5 Display Serial Interface (DSI)

Display Serial Interface (DSI) is a high-speed serial interface based on a number of (1GBits) data lanes. The total voltage swing of the data lines is only 200mV. This makes the electromagnetic noise created and power consumed very low. DSI displays are created for special purposes like for handheld devices. Interfacing of 800x480 pixels, capacitive touch, 7" TFT display via DSI connecter is shown in figure 5.9. This display can be powered using separate power supply or using USB port at Raspberry Pi or using GPIO pins of Raspberry Pi.

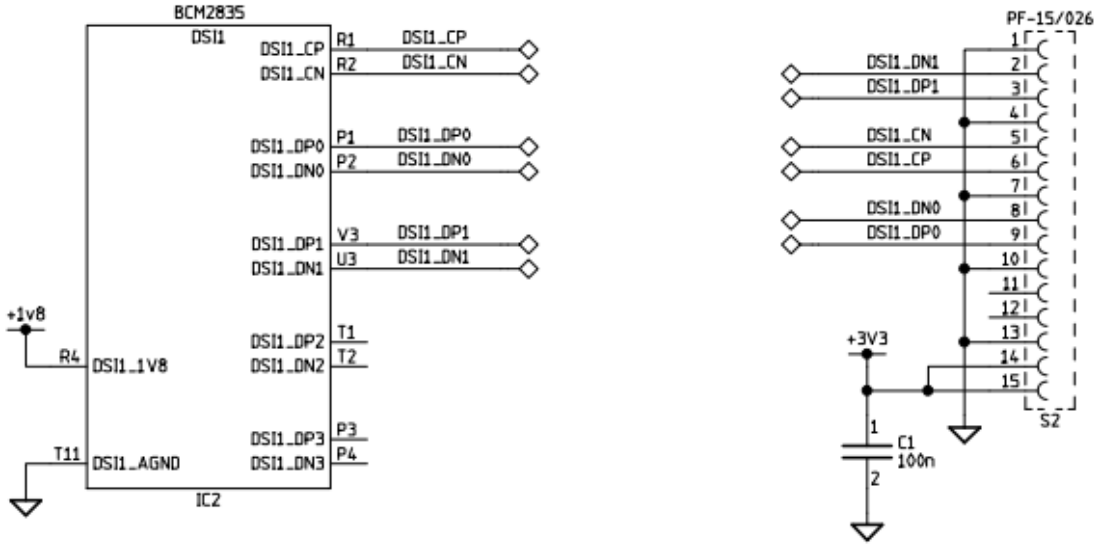


Fig 5.9 Interfacing diagram of DSI connector [6]

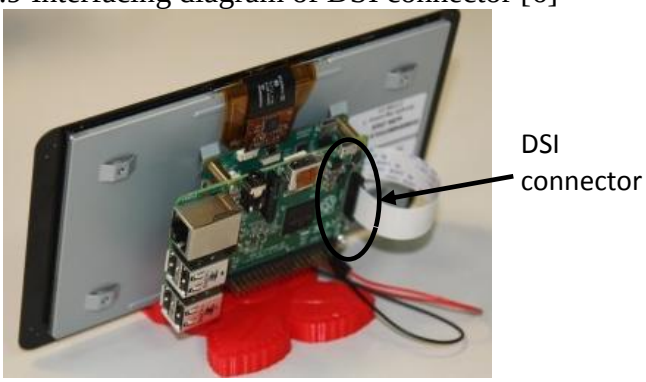


Fig 5.10 DSI Interfacing for TFT [9]

Exercise:

1. Interface camera and take image and video on TFT connected Raspberry Pi
2. Write a program to showcase a mobile phone keypad.
3. Write a program to develop a standard calculator.
4. Write a program for switches on touchpad.

Reference:

[1] www.tinylcd.com
[2] http://www.tinylcd.com/tiny149/product.php?id_product=55
[3] http://www.averagemanvsraspberrypi.com/2014/08/35-touchscreen-tft-display-for.html
[4] http://hackyourmind.org/public/images/display7seg_circuit.png
[5] https://hackster.imgix.net/uploads/image/file/47586/7segmentos5.gif?w=580&h=435&fit=max&s=3756b15ab456edc45ff98c863120cf43
[6] https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/Raspberry-Pi-Rev-2.1-Model-AB-Schematics.pdf
[7]http://sainsmart.tumblr.com/post/52849596106/diagramsainsmart-18-tft-lcd-modules-connect
[8] http://www.tinylcd.com/tiny149/product.php?id_product=55
[9] https://www.raspberrypi.org/wp-content/uploads/2015/08/rear_view.jpg
[10] http://swag.raspberrypi.org/products/raspberry-pi-7-inch-touchscreen-display
[11] http://rpi.science.uoit.ca/lab/ssdispay/

Chapter 6

Camera Interfacing and Programming

6.1 Introduction

Multimedia operation is one of the major component of camera interface. This chapter aims to setup the Raspberry Pi camera module, take pictures & video using debian commands and python code using python Picamera module. The Raspberry Pi board supports the following types of camera modules like Raspberry Pi Camera, Spy Camera and NoIR Camera. The **Raspberry Pi Camera** (Fig 6.1) has a 5MP Omni vision 5647 sensor in a fixed focus module. The module attaches to Raspberry Pi board by way of a 15 Pin Ribbon Cable to the dedicated 15-pin MIPI Camera Serial Interface (CSI) connector on the board which has designed especially for interfacing the cameras. The **Spy Camera** (Fig 6.2) for Raspberry Pi is smaller than a thumbnail with a high enough resolution to see people. It's about the size of 8.5mm x 11.3mm which is similar to cell phone camera. The Raspberry Pi **NoIR Camera** (Fig 6.3) board has no Infrared filter which makes it perfect for taking Infrared photographs or photographing objects in low light (twilight) conditions.

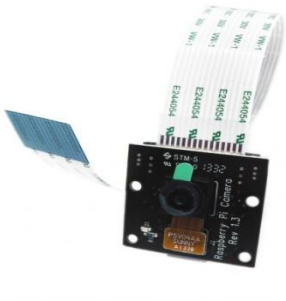


Fig: 6.1 Raspberry Pi Camera[7] Fig: 6.2 Spy Camera[8] Fig: 6.3 NoIR Camera[9]

6.2 Hardware Requirement:

Raspberry Pi compatible camera (Raspberry Pi Camera) is 5 Mega Pixel with FFC-CSI Camera Ribbon Cable of suitable length. In this chapter Raspberry Pi Camera is considered for interfacing, however camera modules like NOIR, spy camera may also be interfaced subject to requirement in future or in next release.

6.2.1 Camera Specifications	
Size, Weight	around 25 x 20 x 9 mm, 3g
Still resolution	5 Megapixels(2592 x 1944 pixels)
Video modes	1080p30, 720p60 and 640x480p60/90
Sensor	OmniVision OV5647
Pixel size	1.4 µm x 1.4 µm
S/N ratio	36 dB
Well capacity	4.3 Ke-
Focal length	3.60 mm +/- 0.01
Horizontal/Vertical view	Horizontal 53.50 degrees, Vertical 41.41 degree
Focal ratio (F-Stop)	2.9

6.2.2 Hardware Features	
Chief ray angle correction	YES
Global and Rolling shutter	Rolling
Automatic Exposure Control, Automatic White Balance, Automatic Black Level Calibration, Automatic 50/60 Hz luminance detection	No- Done by ISP Instead
Frame Rate Upto 120 fps	Max 90 fps Limitations on frame size for the higher frame rates (VGA only for above 47fps)
AEC/AGC 16-zone size/position/weight control	No- Done by ISP Instead
Mirror and flip	YES
Cropping	No- Done by ISP Instead (except 1080p mode)
Lens Correction, defective pixel canceling	No- Done by ISP Instead
10 bit raw r gb data	YES, format conversions available via GPU
MIPI Interface(two lanes)	YES
32 bytes of embedded one time programmable memory	No
Embedded 1.5V regulator for core power	YES

6. 2. 3 Software Features	
Picture formats	JPEG, JPEG+RAW, GIF, BMP, PNG, YUV420, RGB888
Video formats	Raw h.264
Effects	Negative, solarise, postarize, whiteboard, blackboard, sketch, emboss, oilpaint, hatch, denoise, gpen, watercolor, film, blur, saturation
Exposure modes	Auto, night, backlight, spotlight, sports, snow, beach, antishake, fireworks
Metering modes	Average, spot, backlight, matrix
Automatic white balance modes	Off, auto, sun, cloud, shade, flash, horizon
Triggers	Keypress, unix signal, time out
Extra modes	Demo, circular buffer, segmented video

6.3 Camera Interfacing:

Interface the Raspberry Pi Camera module by inserting the camera cable into the Raspberry Pi as shown in figure 6.4. The FFC (Flexible Flat Cable) slots into the connector situated between the Ethernet and HDMI ports (near A/V out) with the silver connectors facing the HDMI port and blue side facing Ethernet port. It is a 15-pin surface mounted flat flexible connector, MIPI Camera Serial Interface 2 (CSI-2) interface connector for camera modules, providing two data lines, one clock line, bidirectional control interface compatible with I2C, 3.3V and GND.

Fig: 6.4 Camera Interfacing [11]

6.3.1 Placing FFC in slot: Lift the tab on the top and place the strip in the connector (blue side facing the Ethernet port) while holding the strip in place, push down the tab.

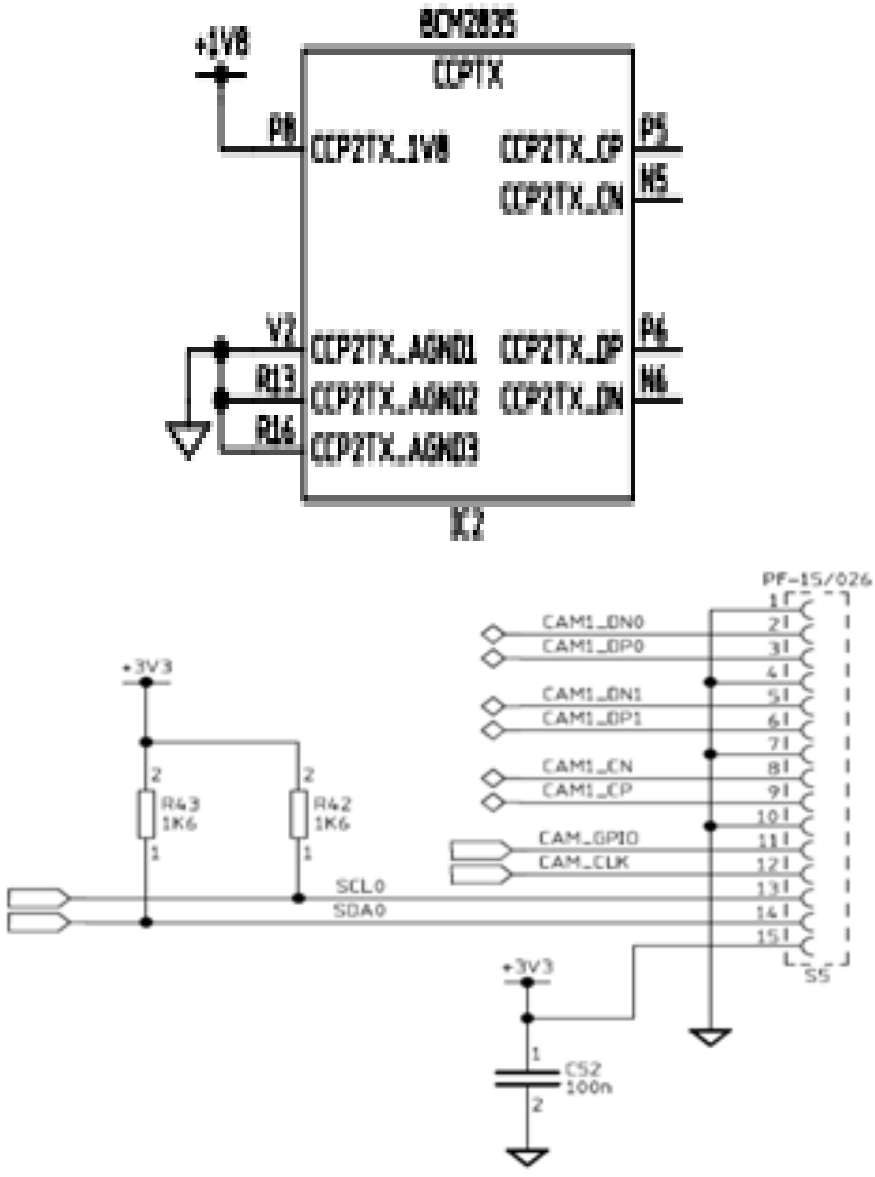


Fig: 6.5 Pin Diagram of camera module [10]

6.4 Camera Configuration: Camera need to be enabled at configuration menu of Raspberry Pi before it is used. This will be done at configuration menu using following steps:

1. Enter $**sudo raspi-config** at terminal
2. At the menu, navigate to Enable Camera, select **Enable**, then select **Finish**
3. **Reboot** Raspberry Pi as shown in screen shots shown in figure 6.6

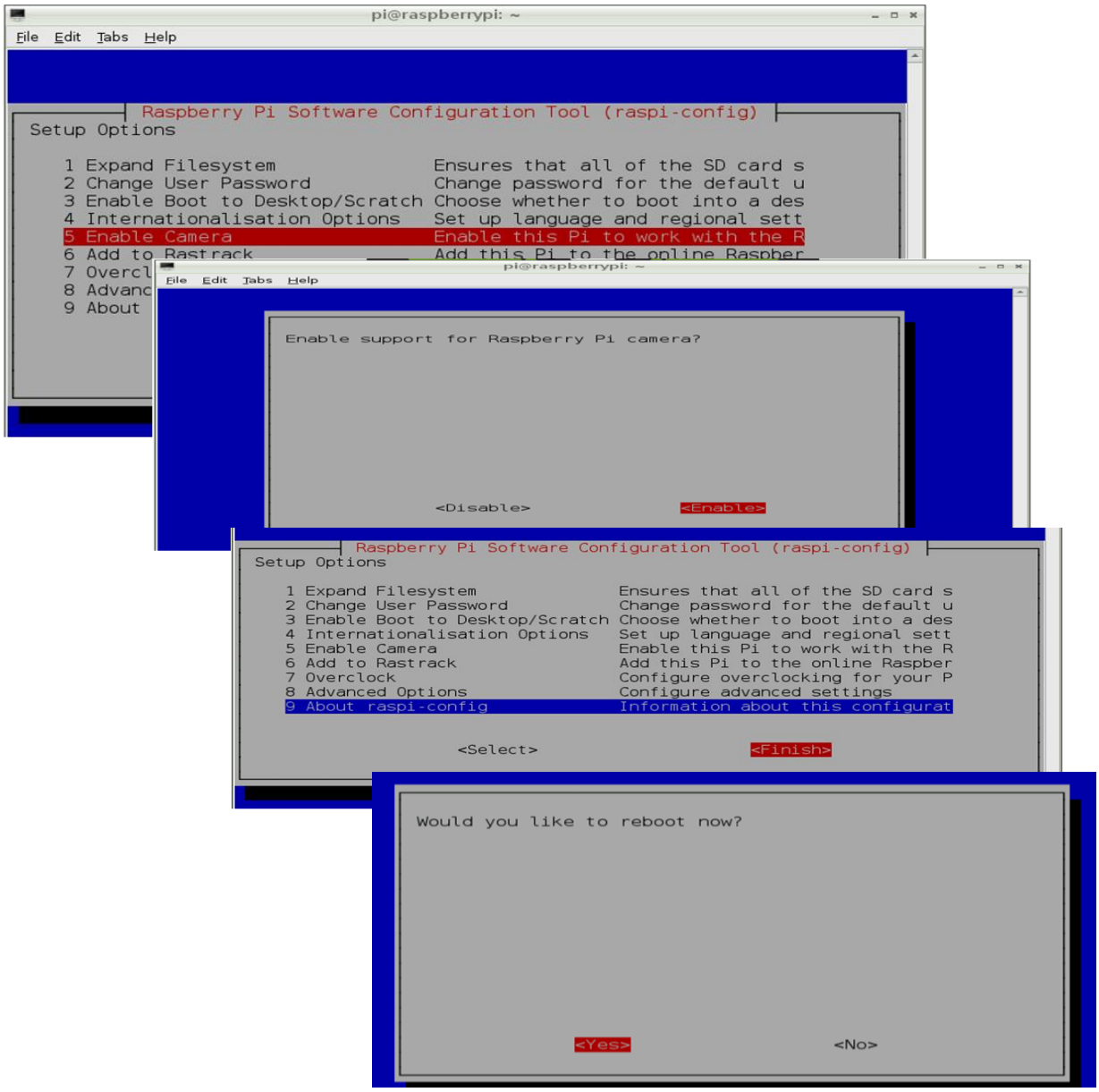


Fig: 6.6 Screen Shots of Camera Interfacing

6.4.1 Camera Configuration Options: After enabling camera at configuration menu, camera features can be used. Power up (Reset) Raspberry Pi, enter username and password. To use camera for capturing images and videos on command line or through python code, the procedure given in next section is followed.

Start by updating , upgrading and installing Python picamera, GPIO library packages and omxplayer with the following commands

1. **$ sudo apt-get update**
2. **$ sudo apt-get upgrade**
3. **$ sudo apt-get install python-picamera python3-picamera python-rpi.gpio**
4. **$ sudo apt-get install omxplayer**

To preview an image for 2 sec on LX Terminal and then store captured image with defined file name by using **$ raspistill –t 2000 -o filename.jpg** command.

Where **2000** represents time in milliseconds and hence 2000 msec or 2 sec.***-o*** is for setting image as output and ***filename.jpg*** is for storing previewed image with name image3 and image type jpg.

To capture some video at the LX Terminal enter **$ raspivid –t 5000 -o filename.h264** command

Where **5000**- for capturing video for 5000msec or 5 sec. **-o** for storing captured video as output file. **filename.h264** for storing video with defined file name and with extension h264.

To Play Captured Video at the LX Terminal enter **$ omxplayer –g filename.h264** command

Example on command line options for camera interfacing.

1. **$ raspistill –p** for taking preview from camera.
2. **$ raspistill –br 30 –o image.jpg** to set brightness level at 30 and saving image output.
3. **$ raspivid –t 5000 –sh 80 –co 70 –br 65 –o video.h264** to set many parameter in single line eg sharpness at 80, contrast at 70 & brightness at 30 and saving video output as .h264 extension file.
4. **$ raspistill –vf –hf –o image.jpg** to set camera rotation. If the camera is placed upside down, the image must be rotated 180° to be displayed correctly.

Note: Stored image may be located at home/pi folder and can be viewed using default image viewer Image magic.

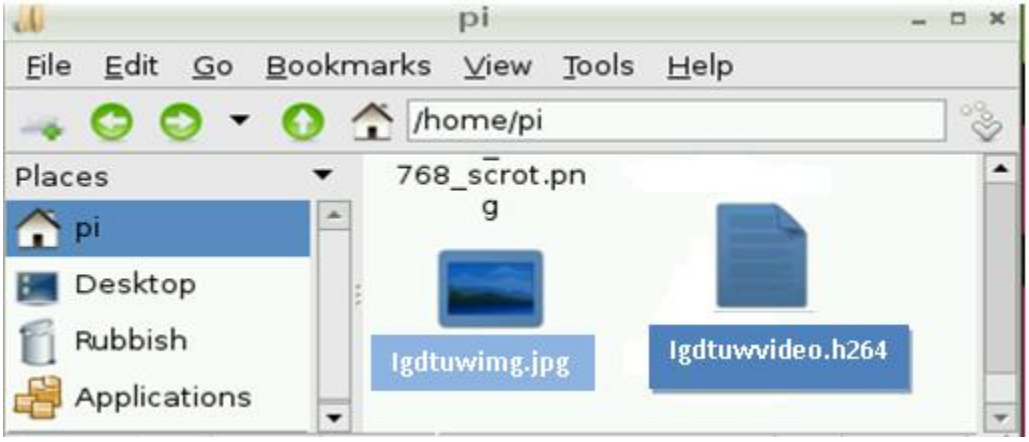


Fig: 6.7 Captured image and video from camera saved at home/pi

6.5 Camera Operations using python programming

Open text editor like nano or leaf pad and write code

import picamera *import time* *camera = picamera.PiCamera()* *camera.capture('imgfile.jpg')* *camera.start_preview()* *camera.vflip = True* *camera.hflip = True* *camera.brightness = 60* *camera.start_recording('videofile.h264')* *time.sleep(15)* *camera.stop_recording()*	 *# To Capture image and save it as imgfile.jpg.* *# To Capture video and save it as videofile.h264.* *#recording video for 15 secs.*

6.6 More on raspistill commands for python codes available in picamera library:

For setting camera modes and effects camera.exposure_mode ='auto' camera.meter_mode ='average' camera.awb_mode ='auto' camera.exposure_compensation =0 camera.video_stablization =False camera.image_effect ='none' camera.color_effects ='none' camera.rotation =0 camera.crop =(0.0, 0.0, 1.0, 1.0)	For setting camera resolution, frame rate and shutter speed camera.resolution =(1024,768) camera.framerate =30 camera.shutter_speed =600000

6.7 Command line extensions with raspistill and raspivid:

-p	Preview window settings <'x,y,w,h'> Allows the user to define the size and location on the screen that the preview window will be placed
-f	Forces the preview window to use the whole screen
-n	Disables the preview window completely
-op	Sets the opacity of the preview windows. 0 = invisible, 255 = fully opaque
-sh	Set image sharpness (-100 to 100), 0 default
-co	Set image contrast (-100 to 100), 0 default
-br	Set image brightness (0 to 100), 50 default, 0 is black, 100 is white
-sa	Set image saturation (-100 to 100), 0 default
-iso	Sets the ISO to be used for captures. Range is 100 to 800

-vs	In video mode only, turn on video stabilization
-ev	Set the EV compensation of the image. Range is -10 to +10, default is 0
-ex	Set exposure mode
-awb	Set Automatic White Balance (AWB) mode
-ifx	Set image effect
-mm	Specify the metering mode used for preview and capture
-rot	Sets the rotation of the image in viewfinder and resulting image (0-359)
-hf	Flips the preview and saved image horizontally
-vf	Flips the preview and saved image vertically
-roi	Set sensor region of interest

Exercise:

1. Write a program to demonstrate different camera settings one by one and save images using these settings.
2. Use camera for automatically capturing images at regular interval.
3. Use camera for capturing video of 5sec duration at regular interval.
4. Write python code for combining above images/videos and playing in one go.

References:

More on camera bash commands:

[1]http://www.raspberrypi.org/documentation/usage/camera/raspicam/raspistill.md.
[2]http://www.raspberrypi.org/documentation/usage/camera/raspicam/raspivid.md.
[3] www.raspberrypi.org/documentation/raspbian/applications/camera.md.
[4] www.Picamera.readthedocs/en/release -1.9/api.html#Picamera.

More on camera specification:

[5]http://www.raspberrypi.org/documentation/hardware/camera.md.
[6] http://www.raspberrypi.org/wp-content/uploads/2013/07/RaspiCam-Documentation.pdf.

More on different cameras:

[7]https://www.modmypi.com/raspberry-pi-camera-board.
[8] http://www.adafruit.com/product/1937.
[9] https:// www.modmypi.com/raspberry-pi-noir-camera-board.
[10] http://www.rs-online.com/designspark/electronics/knowledge-item/r-pi-ffc-connectors.
[11] http://www.adafruit.com/product/1367.

Chapter 7

Audio Interfacing and Programming

7.1 Introduction

Recording and playing audio files is important for the smart device/ mobile device. Raspberry Pi support recording and playing of audio files. This function can lead to many features of an application like accepting voice, commands, playing music etc.

7.2 Interfacing

Playing an Audio file: This can be achieved by three methods:

a) Using alsamixer b) Python programming c) Voice recording

7.2.1 **Using alsamixer**:- Alsamixer is a graphical mixer program for the Advanced Linux Sound Architecture (ALSA) which is used to configure sound settings and adjust the volume [1]. Some alsa-utilities are required for this feature. To download these utilities, run following command

$ sudo apt-get install alsa-utils

1. Playing a .wav file:
   $ aplay file_name.wav

2. Playing a mp3 files:

   Download mp3 player using command
   $ sudo apt-get install mpg321
   $ sudo apt-get install lame
   ***LAME** is a free software codec used to encode/compress audio into the lossy MP3 file format.

   Playing a mp3 file.
   $ mpg321 file_name

Fig 7.1- Playing mp3 file

Note: To end song press: ctrl+c

7.2.2 Using python programming

Second method to play a file in Raspberry Pi is by python programming.

```
import pygame
pygame.mixer.init()
pygame.mixer.music.load("file_name.mp3")
pygame.mixer.music.play()

while pygame.mixer.music.get_busy() == True:
        continue
```

7.3 Using Voice Recording

For voice recording on a Raspberry Pi, a microphone and a USB sound card is required and use following steps:

1. Insert the USB sound card into the USB jack of Raspberry Pi. Now check the module in the directory that the inserted card is recognized by Raspberry Pi or not.
   $ cat /proc/asound/cards

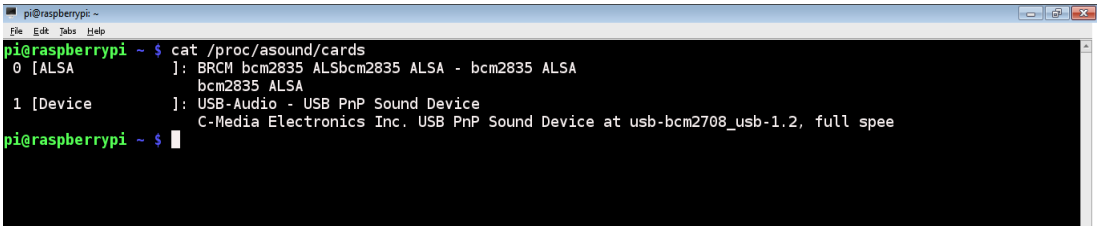


Fig 7.2 Checking Sound card in directory 'card'

2. Check sound module snd_bcm2835 availability by command lsmod.
   If still snd_bcm2835 is not present, load this manually by using command
   $ sudo modprobe snd-bcm2835
3. Open ALSA mixer using command
   $ *alsamixer*
   Choose device ported in USB and here the volume of mic and speaker can be adjusted.
4. Open the editor and write the following line:
   pcm.!default sysdefualt:Device
   and save it as .asoundrc
   This line change the sound settings from default to usb.
5. Now set the sound to USB jack. This can be set to HDMI and audio jack.
   $ amixer cset numid=3 X where X=0 for auto; X=1 for audio jack; X=2 for audio enable through HDMI
6. Now record the .wav file by following command:
   $ arecord –d 5 –r 48000 file_name.wav
   And finally play it by following command:
   $ aplay file_name.wav

Exercise:

1. Record voice and play. Write a program to pause it in between and then play it again.
2. Explore Push-to- On functionality.
3. Write a program to forward and repeat an mp3 file.
4. Explore text-to- speech functionality (E-speak)

References:

1. About alsamixer: http://en.wikipedia.org/wiki/Alsamixer
2. Download the .wav file. These files can be downloaded from the link given below: http://www.freespecialeffects.co.uk/

Chapter 8

GSM Interfacing and Programming

8.1 Introduction

Global System for Mobile Communications, (GSM) is a standard developed by the European Telecommunications Standards Institute (ETSI) to describe protocols for second-generation (2G) digital cellular networks used by mobile phones. GSM supports voice calls and data transfer speeds of up to 9.6 kbps, together with the transmission of SMS (Short Message Service). The GSM module is shown in figure 8.1.

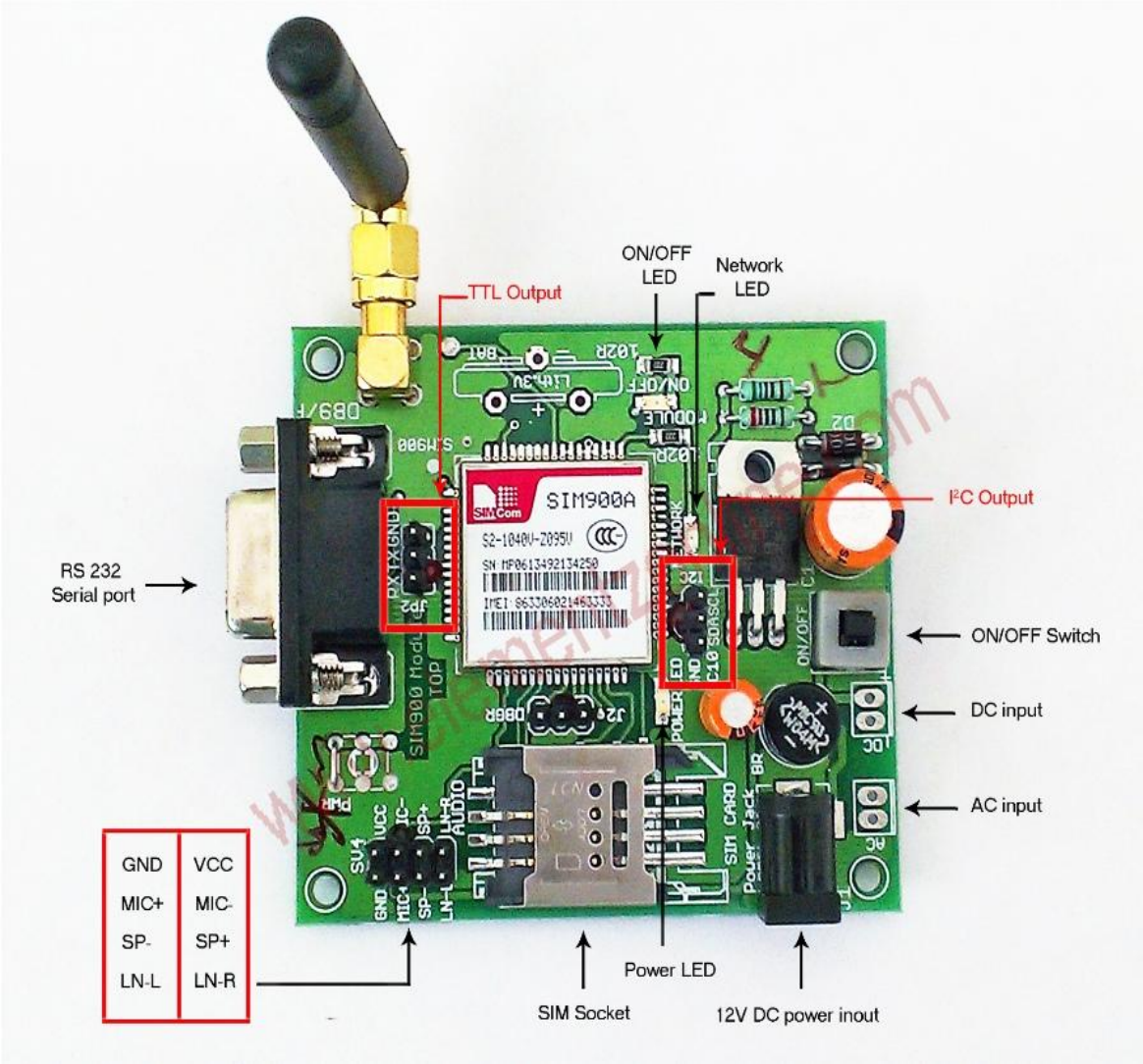


Fig.8.1: GSM Module, SIM900

8.2 Interfacing of GSM Module with Raspberry Pi is given in figure 8.2.

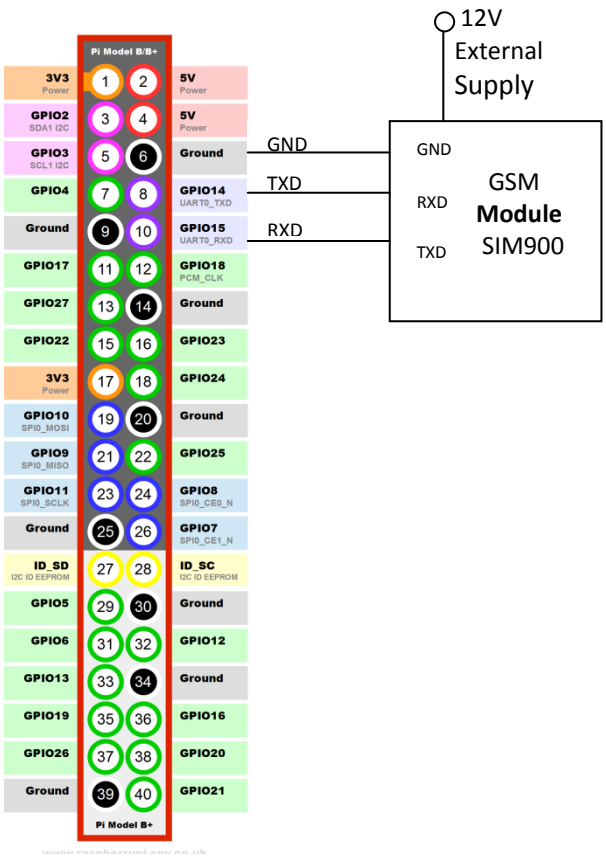


Fig.8.2: Interfacing of GSM Module with Raspberry Pi2

8.2.1 Configuring serial port of Raspberry Pi2

Raspberry Pi2 GPIOs has only one UART port available which includes the TX at pin 8 and Rx at pin 10. By default, UART of raspberry pi is configured for console messages to run Pi using serial console. To use serial port, console settings need to be changed. The UART configuration of the serial port settings are as follows:

To free serial port from console messages, Open LX Terminal and type

$ sudo nano /etc/inittab followed by enter.
Comment out or delete following line by appending '#' before the line
To:23:respawn:/sbin/getty –L ttyAMA0 115200 vt100
Save file with ctrl X followed by Y

i. **To stop debug messages while Pi is booting, edit the cmdline.txt file:**
On LX Terminal and type
$ sudo nano /boot/cmdline.txt followed by enter you will find the following line:

"dwc_otg.lpm_enable=0 console=ttyAMA0,115200kgdboc=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait"
Replace the above text by the text as given below.
"dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait" and Save file with ctrl X followed by Y and **Reboot** the system

After changing configuration settings for UART port and rebooting the pi, some packages and libraries are to be installed as given in following section.

8.2.2 UART package installations

i) Install serial UART library using:

$ sudo apt-get install python-serial

ii) Installation of console applications (minicom or cutecom)

$ sudo apt-get install minicom, to install minicom

Minicom is advantageous in a sense that it works on command line and no graphical interface is required. Different type of files like image, video, documents etc can be handled on UART port using minicom.

$ sudo apt-get install cutecom (Install cutecom)

Cutecom is mainly used for operations involving only commands and is easy to start operating.
Note: for more on minicom /cutecom settings please check reference [6].

8.3 GSM programming there are two ways to program GSM module based on 1) AT commands and 2) Python programming

8.3.1 GSM programming Method 1: Using AT Commands

AT commands are instructions used to control a modem. AT is the abbreviation for attention. Every command line starts with AT.
Functionally AT commands are split into five types

1. General commands:-Includes model name, revision, IMEI, identification etc.
2. Call Control Commands: -Includes dialing, hanging up a call, call parameter setting, call mode setting, Bearer service setting etc.
3. Network service related commands:- Includes network status information, facility lock, phone locking, CLIP, call forwarding/blocking etc
4. Mobile equipment control and status commands:- Includes phone activity status, battery condition , signal quality
5. Mobile equipment errors: -Includes error messages.

Table1: AT Commands for Calling

ATD <Number>;	To dial a number
ATA	To attend a call
AT DL	To redial the last number
ATH	To Hold the call

Table2: AT commands for Messaging

AT+CMGF=1	To enter text mode (to be done once initially)
AT+CMGS = "Recipient Number" then Enter >Message (Then press Crtl+Z)	To send a message
AT+ CMGR= N	To read N^{th} message
AT+CMGD = N	To delete N^{th} message

To run AT command , use any serial console.

Note: AT commands are not case sensitive.

8.3.2 GSM programming: Using Python Code

1. Open new file in leafpad editor and write the python code below.
2. Save file by the name **filename.py**.
3. Run the file in LX Terminal using command **sudo python filename.py**

This sample code is used to test the calling functionality of GSM module using AT Commands

```
import serial
import time
port = serial.Serial("/dev/ttyAMA0", 9600, timeout=3.0)
try:
        print"GSM SIM900\n"
        print"AT          to check operations"
        port.write('AT\r\n')
        rcv = port.read(20)
        print"GSM Working: "+rcv
        while True:
                rcv = port.read(20)
                #print rcv
                #time.sleep(2)
                keyin = raw_input("Want to call ? press y else n\n")
                if (keyin=='y'):
                        keyin = raw_input("Dial Number: ")
                        #print keyin
                        keyin2 = 'ATD '+keyin+';\r\n'
                        print"Dialing : " + keyin
                        port.write(keyin2)
                        time.sleep(2)
                        keyin = raw_input("Type ATH or ath to cut call\n")
                        port.write(keyin+'\r\n')
```

```
                        port.flushInput()
                        port.flushOutput()
                else :
                        if keyin=="n"or"N":
                                rcv= port.read(50)
                                print rcv
                                time.sleep(2)
                if 'RING' in rcv:
                        keyin=raw_input( 'Someone calling\n type ATA to accept')
                        port.write(keyin+'\r\n')
                if '+CMTI:' in rcv:
                        print "New Message received"
                port.flushInput()
                port.flushOutput()
                time.sleep(2)
except:
        port.close()
```

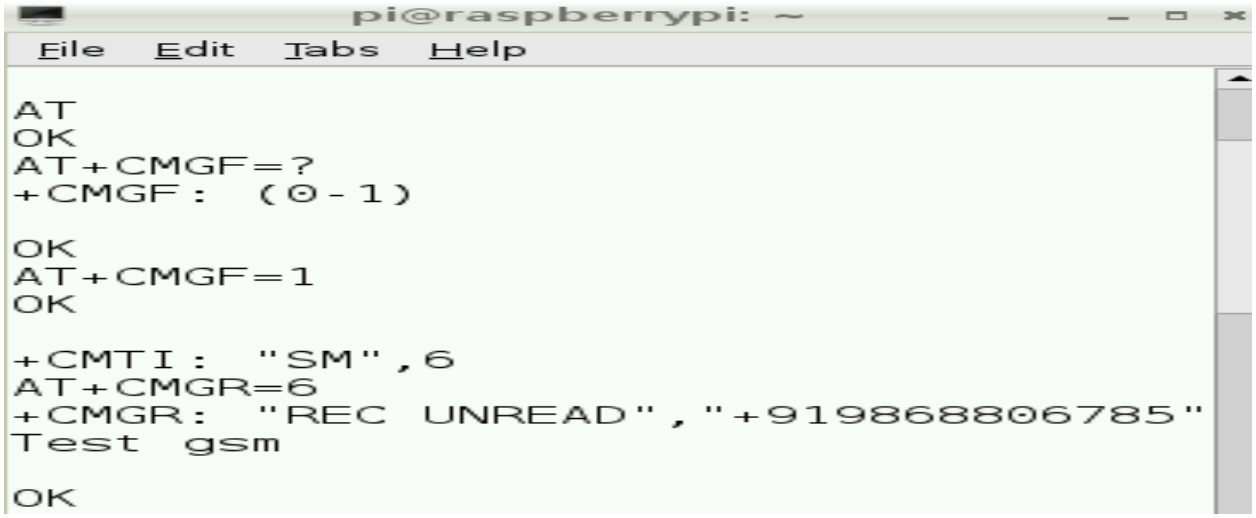
8.4 Dealing with Short Message Service (SMS)

Messaging is an important feature of GSM interfacing. Essential AT commands for dealing with SMS are as per table 2.

Procedure

1. On LXTerminal of Raspberry pi, open serial terminal like minicom.
2. Write AT command for Text Mode setting as
    a. **AT+CMGF=1**
    b. Ok should be received
3. To Read any message stored on SIM
    a. AT+CMGR=N ; For reading N^{th} message
    b. Message will be displayed including sending details
4. To Send any message
    a. AT+CMGS=”Recipient mobile number” , then press enter and enter message to be sent
    b. Press CRTL+Z to end and to send message.
5. To Delete any message
    a. AT+CMGD=N ; For deleting N^{th} message
6. To Write and store any message memory
    a. AT+CMGW= message
    b. Press CRTL+Z to end and to save message.
    c. On successful storage of message, location number of message will be reverted back on screen by +CMGW: N, where N is any number.

Output



8.5 Sending SMS through python code

1. Open new file in leafpad editor and write the python code below.
2. Save file by the name **filename.py**.
3. Run the file in LX Terminal using command **sudo python filename.py**

This sample code is used to test the SMS sending functionality of GSM module using AT Commands.

```
import Adafruit_DHT
import serial
import time
sensor = Adafruit_DHT.DHT11
pin = 22
port= serial.Serial("/dev/ttyAMA0",9600,timeout=3.0)
try:
      print =”Temp=28.0*C  Humidity=57.0%”
      print "Sending message ..."
      msg1= 28.0*C 57.0%”
      number=raw_input("Enter mobile number:")
      port.write('AT+CMGS=”'+number+'”\r\n')
      time.sleep(2)
      port.flushInput()
      port.flushOutput()
      port.write(msg1)
      time.sleep(2)
      port.write('\x1A\r\n')
      print port.read(50)
      port.flushInput()
      port.flushOutput()
except:
      port.close()
```

Output : SMS sending output

8.6 Receiving and printing SMS through python code

1. Open new file in leafpad editor and write the python code below.
2. Save file by the name **filename.py**.
3. Run the file in LX Terminal using command **sudo python filename.py**

This sample code is used to test the SMS receiving functionality of GSM module using AT Commands.

```
import serial
import time
port = serial.Serial("/dev/ttyAMA0",9600,timeout=3.0)

while True:
      port.flushInput()
      rcv=port.read(100)
      print(rcv)

      print "Checking for new messages..."
      if '+CMTI:' in rcv:
            print "msg received"
            msgnum=rcv[14:16]
            port.write('AT+CMGR=' + str(msgnum) + '\r\n')
            port.flushInput()

            time.sleep(1)
            response = port.read(200)
            msg = response[response.index('\n')+1:]
            msg1 = msg[msg.index('\n')+1:]
            msg2 = msg1[msg1.index('\n')+1:]
            msg3 = msg2[msg2.index('\n')+1:]
            res = response[response.index(msg2):(response.index(msg3)-
1)]
            print res[0:6]
            print res[6:11]
      else:
            print "No new message"
```

Output : SMS receiving output

Exercise:

1. Write a program to demonstrate different AT Commands for GSM one by one and save images using these settings.
2. Use AT commands for message sending and receiving.
3. Use AT commands for GPRS working on GSM module at regular interval.
4. Write python code for combining above all functionalities in one go.

References:

1. http://www.propox.com/download/docs/SIM900_AT.pdf
2. http://www.hobbytronics.co.uk/raspberry-pi-serial-port
3. http://elinux.org/RPi_Serial_Connection
4. http://www.raspberrypi-spy.co.uk/2013/12/free-your-raspberry-pi-serial-port/
5. http://home.budget.net/~klricks/Comp/RPi/Serial.html
6. https://help.ubuntu.com/community/Cutecom

Chapter 9

GPS Interfacing and Programming

9.1 Introduction

A Global Positioning System (GPS) is a satellite based navigational system designed to navigate on the earth, air and into water. It provides location and time information in all weather conditions with the help of GPS satellite system.

GPS receiver can be used worldwide to find three-dimensional location (latitude, longitude, and altitude) and precise timings using many channel satellite broadcast signals from space. The GPS receiver's tracking sensitivity allows continuous position coverage in nearly all application environments. National Marine Electronics Association's NMEA 0183 v3.0 protocol supports baud rate of 9600 or 115200 bps. GPS receivers are widely used in smart devices for vehicle and animal tracking etc. two available models of GPS receivers are shown below.

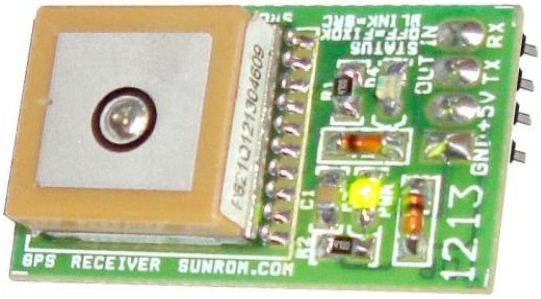


Fig 9.1(a): GPS Receiver [5]

Fig 9.1(b): GPS Receiver [7]

9. 2 GPS Module Working

A GPS receiver calculates its position using microwave signal from constellation of 24 and 32 Medium Earth Orbit satellites. Each satellite continually transmits messages containing the time the message was sent, a precise orbit for the satellite sending the message in a specific format.

Signal information is used to estimate the position of the GPS receiver as the intersection of sphere surfaces. The resulting coordinates are converted to provide latitude and longitude or location on a map. At least four satellites are needed to know position and to correct the GPS receiver clock as three sphere surfaces intersect in two points.

9. 3 Block Diagram of GPS Operation

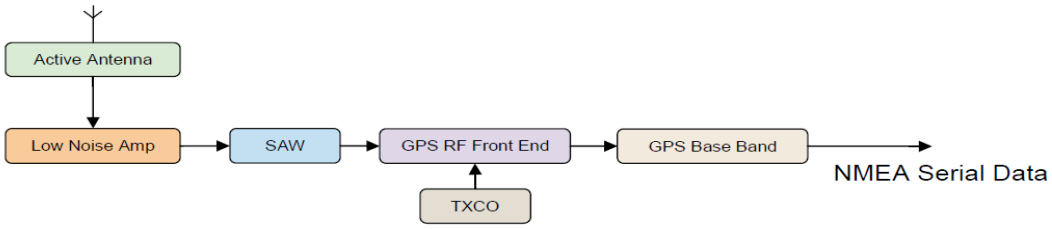


Fig9.2: Block diagram of GPS operation [6]

The GPS Receiver consists of two units, first is active antenna that takes power from the module and amplifies the received RF signals for high sensitivity. The RF signal is filtered and processed to generate NMEA format serial data output.

Interfacing Diagram: The interfacing diagram of GPS receiver with Raspberry PI2 is shown in fig 9.2. It is interfaced with the help of UART communication.

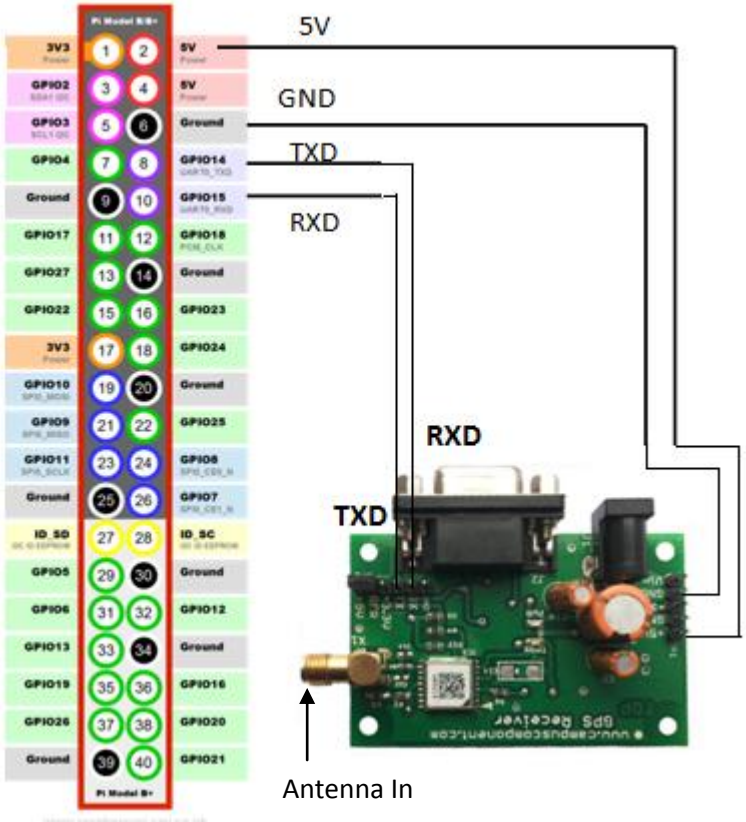


Fig9.3: Interface diagram of GPS receiver

***Note: Remember to connect Antenna with GPS module**

9.4 NMEA Messages and Protocol

The serial interface protocol is based on the National Marine Electronics Association's NMEA 0183 that defines communication interface and the data format. The GPS receiver supports the latest release of NMEA 0183, Version 3.0. All NMEA-0183 begin with $ and end with a carriage return and a line feed. Data fields follow comma (,) delimiters and are variable in length. Null fields still follow comma (,) delimiters, but contain no information. An asterisk (*) delimiter and checksum value follow the last field of data contained in an NMEA-0183 message. The checksum is the 8-bit exclusive of all characters in the message, including the commas among fields, but not including the $ and asterisk delimiters. The hexadecimal result is converted to two ASCII characters (0–9, A–F). The most significant character appears first. The following table 9.1, summarizes the set of NMEA messages supported by the receiver.

The entire protocol encompasses over 50 messages, only a sub-set of these messages apply to this any GPS receiver. Some of standard NMEA 0183 protocol used are:

1. **$GPGGA** - Global Positioning System Fix Data
2. **$GPGLL** - Geographic position, latitude / longitude
3. **$GPRMC** - Recommended minimum specific GPS/Transit data
4. **$GPVTG** - Track made good and ground speed

NMEA 0183 Message Format Description

$IDMSG,D1,D2,D3,D4,.....,Dn*CS[CR][LF]

"$"	**The "$" signifies the start of a message.**
ID	The identification is a two letter mnemonic which describes the source of the navigation information. The GP identification signifies a GPS source.
MSG	The message identification is a three letter mnemonic which describes the message content its number and order of the data fields
","	Commas serve as delimiters for the data fields.
Dn	Each message contains multiple data fields (Dn) which are delimited by commas. The length of the fields can be variable.
"*"	The asterisk serves as a checksum delimiter.
CS	The checksum field contains two ASCII characters which indicate the hexadecimal value of the checksum.
[CR][LF]	The carriage return [CR] and line feed [LF] combination terminate the message.

Table 9.1. NMEA 0183 Message Format Description

NMEA 0183 standard messages vary in length, but each message is limited to 79 characters or less. This length limitation excludes the "$" and the [CR][LF]. The standard message data field block, including delimiters, is limited to 74 characters or less.

Example of NEMA message format

GGA - Global Positioning System Fix Data, Time, position and fix related data for a GPS receiver. The structure for the same is shown below

$GPGGA,hhmmss.sss,ddmm.mmmm,a,dddmm.mmmm,a,x,xx,x.x,x.x,M,,,,xxxx*hh
1 2 3 4 5 6 7 8 9 10 11

<*CR*><*LF*>Field

Example:

*$GPGGA,111636.932,2447.0949,N,12100.5223,E,1,11,0.8,118.2,M,,,,0000*02<CR><LF>*

GPS Raw Data as received from GPS Receiver at Serial Console Minicom / Cutecom

```
$GPGGA,105124.000,2839.8761,N,07713.9209,E,1,6,1.41,223.7,M,-35.9,M,,*73
$GPGSA,A,3,02,26,05,18,29,21,,,,,,,1.66,1.41,0.87*0A
$GPRMC,105124.000,A,2839.8761,N,07713.9209,E,0.47,286.83,230914,,,A*6C
$GPVTG,286.83,T,,M,0.47,N,0.87,K,A*36
$GPGGA,105125.000,2839.8761,N,07713.9207,E,1,7,1.20,223.7,M,-35.9,M,,*7A
$GPGSA,A,3,02,26,05,15,18,29,21,,,,,,1.48,1.20,0.86*04
$GPRMC,105125.000,A,2839.8761,N,07713.9207,E,0.50,286.83,230914,,,A*65
$GPVTG,286.83,T,,M,0.50,N,0.93,K,A*35
```

Fig9.4: Raw data in NEMA format as received at serial port

Note: - *Required connection diagram and settings for serial port of Raspberry Pi will be done same as described earlier for GSM at chapter 8.

9. 5 General GPS Receiver User's Tips

If the satellite signals cannot be locked or experiencing receiving problem (while in urban area), following steps are used:

a) Relocate the GPS receiver near window or open sky for better receiving performance.

b) Move to another open space or reposition GPS receiver toward the direction with least blockage.

c) Move the GPS receiver away from the interference sources.

d) Wait until the weather condition is improved.

Limitations

a) Some vehicles having heavy metallic sun protecting coating on windshields may affect signal receptions

b) Driving in and around high buildings, and tunnels may affect signal reception.

c) When GPS receiver is moving, it will take longer time to get its position fixed. Wait for satellite signals to be locked at a fixed point when first power-on the GPS receiver to ensure quick GPS position fix

d) Weather will affect GPS reception – rain & snow contribute to worsen sensitivity.

Python code for extracting latitude, longitude, speed and course from NEMA message

```
#!/usr/bin/python
import serial

port = serial.Serial("/dev/ttyAMA0", 9600, timeout=3.0)
while True:
        rcvdfile=port.read(1200)
        pos1 = rcvdfile.find("$GPRMC")
        pos2 = rcvdfile.find("\n",pos1)
        loc = rcvdfile[pos1:pos2]
        data = loc.split(',')

        pos11 = rcvdfile.find("$GPGGA")
        pos22 = rcvdfile.find("\n",pos11)
        loc1 = rcvdfile[pos11:pos22]
        data1 = loc1.split(',')
        if data[2]=='V':
                print 'No location found'
        else:
                #print "UTC time="+data[1]+" UTC date=" + data[9]
                gps_time=float(data[1])
                gps_date=float(data[9])

                gps_hour=int(gps_time/10000.0)
                gps_min= gps_time%10000.0
                gps_sec=gps_min%100.0
                gps_min=int(gps_min/100.0)
                gps_sec=int(gps_sec)
                gps_dd=int(gps_date/10000.0)
```

```
gps_mm= gps_date%10000.0
gps_yy=gps_mm%100.0
gps_mm=int(gps_mm/100.0)
gps_yy=int(gps_yy)

print 'time=',gps_hour,':',gps_min,':',gps_sec
print 'date=',gps_dd,'/',gps_mm,'/',gps_yy

print "Latitude = "+data[3]+data[4]
print "Longitude = "+data[5]+data[6]
print "Speed = "+data[7]
print "Course = "+data[8]
print "\n"
gps_time_gga=float(data1[1])
gps_hour_gga=int(gps_time_gga/10000.0)
gps_min_gga= gps_time_gga%10000.0
gps_sec_gga=gps_min_gga%100.0
gps_min_gga=int(gps_min_gga/100.0)
gps_sec_gga=int(gps_sec_gga)

print ‘time=',gps_hour_gga,':',gps_min_gga,':',gps_sec_gga
print "Latitude = "+data1[2]+data1[3]
print "Longitude = "+data1[4]+data1[5]
print "Satellites used= "+data1[7]
print "Altitude = "+data1[9]
print "\n"
```

Output

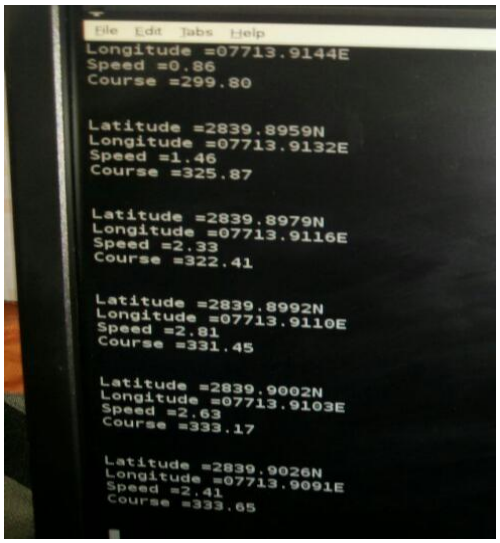


Fig 9.5 (a) GPS data on minicom, depicts the GPS location of IGDTUW as mentioned in fig 9.5

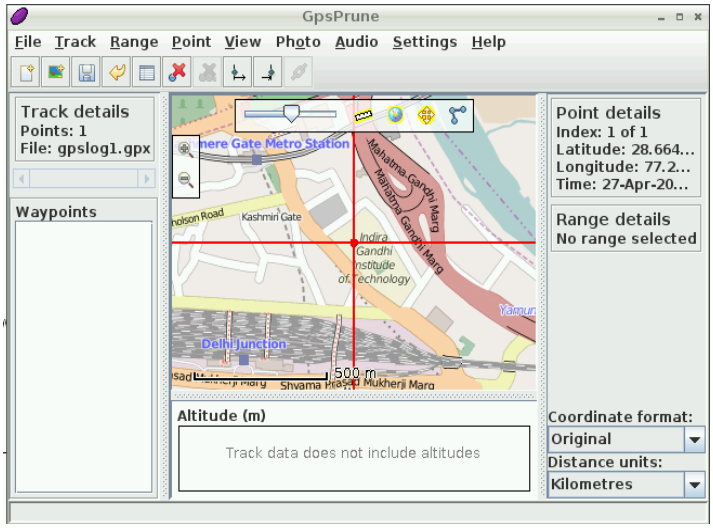


Fig 9.5(b) GPS location at Map

Exercise:

1. Write a program to extract longitude, latitude and altitude one by one and save images using these settings.
2. Write a program to extract course and speed.
3. Write a program to extract location information and plot it on gpsprune.
4. Write python code for combining above all functionalities in one go.

References:

1. http://aprs.gids.nl/nmea/
2. http://www.trimble.com/OEM_ReceiverHelp/V4.44/en/NMEA-0183messages_MessageOverview.html
3. http://www.gpsinformation.org/dale/nmea.htm
4. NMEA 0183, Version 3.01 standard may be obtained from NMEA, www.nmea.org
5. http://sunrom.com/p/gps-receiver-ttl-uart-patch-antenna-on-top
6. http://sunrom.com/media/files/p/225/1213-datasheet.pdf
7. http://3.imimg.com/data3/CY/IP/MY-3287828/sim28m_gps-modem-500x500.png

Chapter 10

Bluetooth and Wi-Fi Interfacing

10.1 Introduction

Bluetooth is a name given to a wireless technology standard for exchanging data over short distances using short-wavelength Ultra High Frequency (UHF) radio waves in the industrial, scientific and medical (ISM) band from 2.4 to 2.485 GHz from fixed and mobile devices, and building Personal area networks (PANs) invented by telecom vendor Ericsson in 1994. IEEE standardized Bluetooth as IEEE 802.15.1. Bluetooth communicates over a range of 0-30 feet (10 meters) with power consumption of 0 dBm (1mW) and its range can be increased to 100 meters by amplifying power.

Bluetooth is a technology allowing wireless transmission of data between two Bluetooth capable devices such as- notebooks, laptops, personal computers, mobile phones and other electronic devices. It makes use of Frequency Hopping Spread Spectrum Technique (FHSS) in order to avoid any interference. Bluetooth communication occurs between a master and slave radio and each radio is capable of operating as a master and slave. Each radio consists of fixed unique 48-bit device address. Two or more Bluetooth radio devices form an ad-hoc network called as PICONET as shown in figure 10.1. Each piconet consists of one master and many slaves. There may be up to seven active slaves at a time in one piconet. In a piconet, master initiates a Bluetooth communication link. Once, a link is established the slave can request a master and it uses the statistical time division multiplexing for effective channel utilization

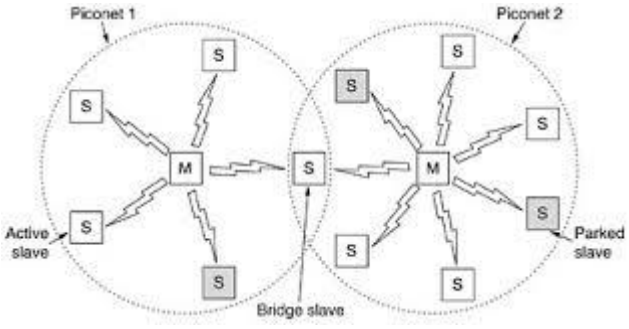


Fig 10.1. Bluetooth Architecture

Fig 10.2. HC-05 Bluetooth Module

10.2 Software and Hardware Requirements for Bluetooth interfacing and programming

Bluetooth SPP Manager – Android Mobile App [6]

10.2.1 Hardware Requirements

HC-05 Module is required HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. The Bluetooth module works as a Serial (RX/TX) pipe i.e. utilizes Serial Port Profile. Any serial stream from 9600 to 115200bps can be transferred seamlessly to the target.

10.2.2 Bluetooth Specifications[3]

a) Bluetooth protocol: Bluetooth specification V2.0+EDR
b) Frequency: 2.4 GHz ISM band
c) Modulation : GFSK
d) Emission power: <= 4dBm, class 2
e) Sensitivity: <=-84dBm at 0.1% BER
f) Speed: Asynchronous: 2.1 Mbps (Max)/160 Kbps & Synchronous: 1 Mbps/1Mbps
g) Security: Authentication and encryption
h) Profiles: Bluetooth serial port
i) Power supply : +3.3VDC 50mA

10.3 Bluetooth Communication between ANDROID Device & Raspberry Pi via Serial UART

Interfacing of HC-05 Bluetooth module to Raspberry Pi and establishing a communication between an android based mobile phone and Raspberry Pi.

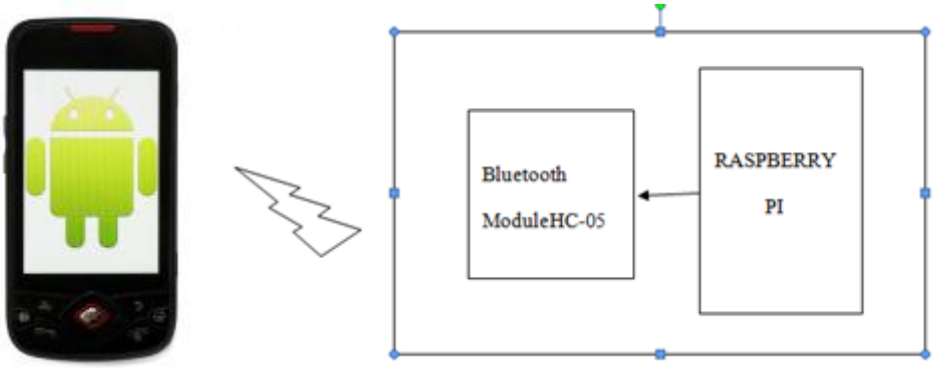


Fig.10.3 : HC-05 Bluetooth Module connection with android device

Procedure

1. Connect the hardware as shown in Fig 10.4 .
2. Install the required packages and configure the serial port as discussed in chapter 8.
3. Open new file in leafpad editor and write code in Python
4. Save file by the name **serialtest.py**.
5. Run the file in LX Terminal using command **"sudo python serialtest.py"**

***Note:-** *Required connection diagram and settings for serial port of raspberry pi will be done same as described earlier for GSM in chapter 8*

10.3.1 Python code for testing serialtest.py

```
import serial                                          # to access serial port
import time                                            #to access time functionality
port=serial.Serial("/dev/ttyAMA0",9600, timeout=3.0)   # opening serial port ttyAMA0
print"\nBluetooth Communication Between RPI &
ANDROID mobile Device\n"
print "Send somthing from Android device"
while True:
        port.flushInput()
        port.flushOutput()

        time.sleep(2)
        rcv=port.read(50)
        if rcv:                                        # read if text available at serial
                print"Received String from Android: " + rcv   port up to length of 50
                time.sleep(2)                          # to give delay of 5 ms
                keyin = raw_input("Enter a string to send:")   # asking for input from keyboard

                port.write(keyin)
                print "Send somthing from Android device"
```

10.3.2 Interfacing diagram for Bluetooth module with Raspberry Pi

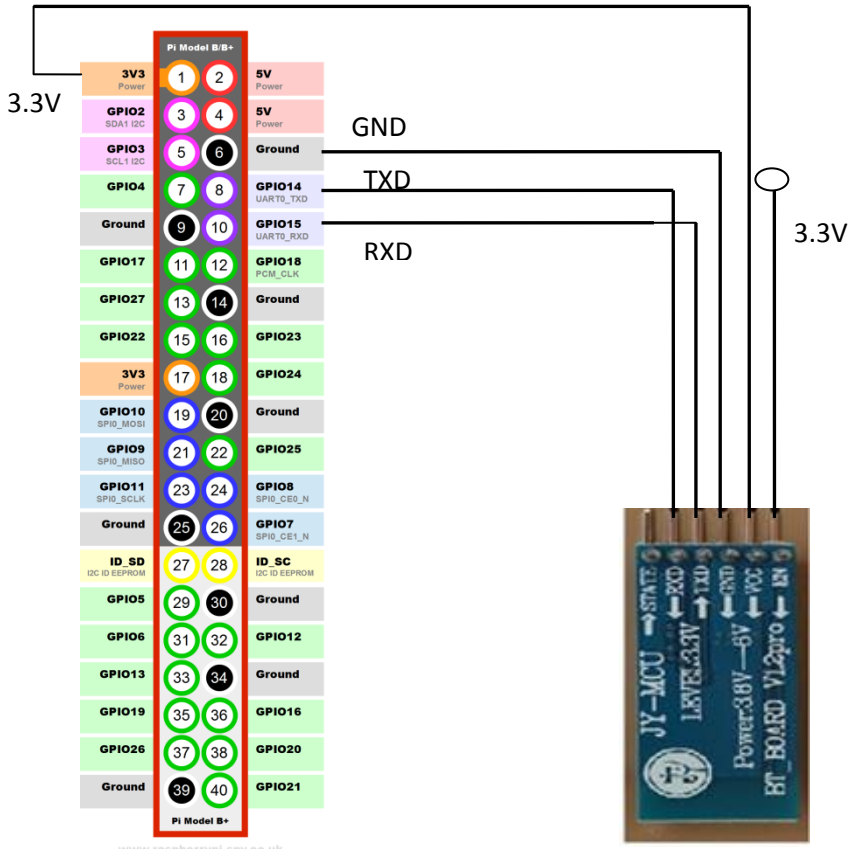


Fig.10.4: HC-05 Bluetooth Module connection with Raspberry Pi

10.3.3 Output

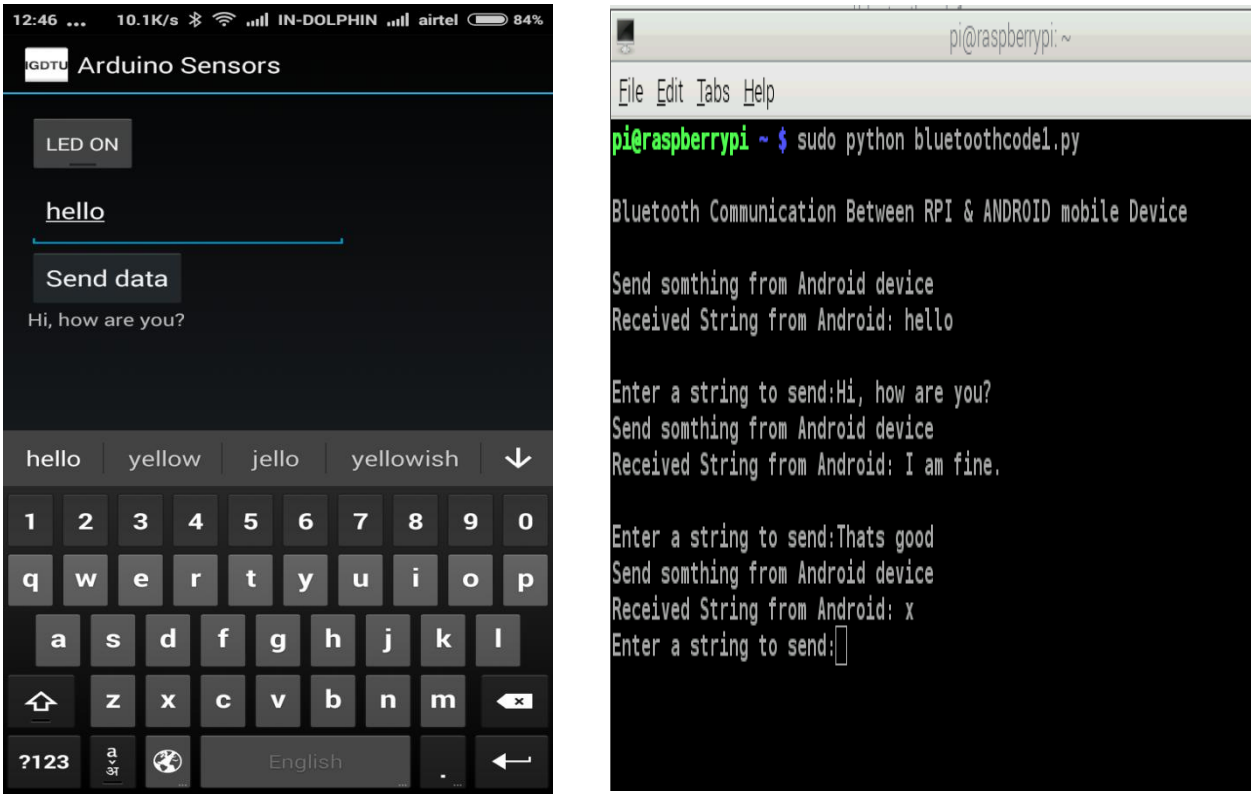


Fig.10.5 Sending & Receiving message between RPi and Android device using My_BT_App

10.4 AT Command Mode

AT is short form of ATTENTION. AT commands are instructions which are implemented to control any modem such as GSM, GPRS, and BLUETOOTH etc. Ex: as shown in table below:

Description	Command	Respond	Parameter
Test	AT	OK	-
Reset	AT+RESET	OK	-
Get module address	AT+ADDR?	+ADDR:<Param> OK	Param: address of Bluetooth module
Set/check module name	AT+NAME=<Param> AT+NAME?	OK +NAME:<Param> OK (/FAIL)	Param: Bluetooth module name (Default :HC-05)
Set/check module mode	AT+ROLE=<Param> AT+ ROLE?	OK +ROLE:<Param> OK	Param: 0-Slave 1-Master 2-Slave-Loop
Set/check query access patterns	AT+INQM=<Param>,<Param2>, <Param3> AT+ INQM?	1.OK 2. FAIL +INQM：<Param>,<Param2>, <Param3>	Param1: 0-inquiry_mode_standard 1-inquiry_mode_rssi Param2: Maximum number of Bluetooth devices to respond to Param3: Timeout (1-48 : 1.28s to

		OK	61.44s) Example: AT+INQM=1,9,48\r\n OK AT+INQM\r\n +INQM:1, 9, 48 OK
Set/check connect mode	AT+CMODE=<Param> AT+ CMODE?	OK +CMODE:<Param> OK	Param: 0 - connect fixed address 1 - connect any address 2 - slave-Loop
Set/check fixed address	AT+BIND=<Param> AT+ BIND?	OK + BIND:<Param> OK	Param: Fixed address (Default 00:00:00:00:00:00) Example: AT+BIND=1234，56，abcdef\r\n OK AT+BIND?\r\n +BIND:1234:56:abcdef OK

Table.10.1 AT Commands and its description

10.5 Setting up AT COMMAND MODE in Bluetooth module

1. After Connecting all the Bluetooth connections with RPi , make HC05 Pin no. 34 i.e., Key pin high by giving 3.3V Supply.
2. Now configure the module for desired operations.

10.5.1 Bluetooth Communication between two Raspberry Pi's via Serial UART

Prepare two set ups of Bluetooth module with Raspberry Pi UART as discussed above and shown in fig 10.6.

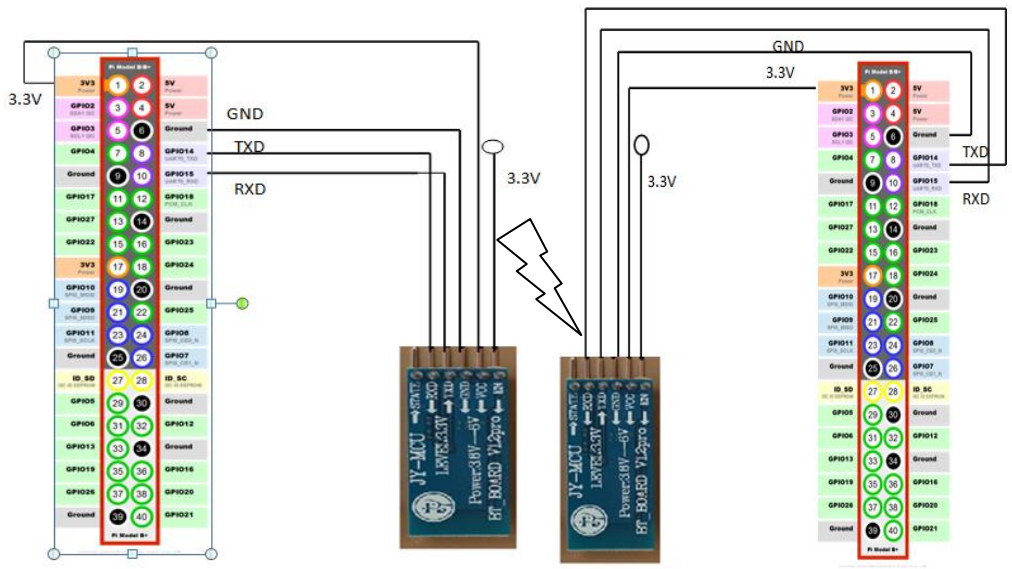


Fig.10.6 Communication between two Raspberry Pi's using two Bluetooth modules

10.5.2 Master/Server Side code to Transmit JPG File

The code starts by importing necessary packages and using GPIO 25 to which key pin of Bluetooth HC05 is attached. The basic code flow:

1. Setting high key pin
2. Setting the module as master
3. Searching nearby Bluetooth Devices
4. Pairing with desired Bluetooth module
5. Asking user any jpg file name which is stored at default location /home/pi

```
import serial
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(25, GPIO.OUT)
port = serial.Serial("/dev/ttyAMA0",9800,
timeout=3.0)
try:
GPIO.output(25 , GPIO.HIGH)
print"Bluetooth\n"
print "List of operating Commands"
                print " Commands functions "
                print " AT+ROLE = 1 to set
        Master "
                print " AT+ROLE = 0 to set
        Slave "
print " AT+INQ to search "
print " AT+BIND=<addr(1234,56,abcdef)>
        to connect to specified address”
print " AT+PAIR=<addr,time> to pair with
remote BT device”
                port.write('AT+ROLE=1\r\n')
rcv = port.read(20)
print "Bluetooth Role " + rcv
                port.write('AT+INQM =
        1,9,48\r\n')
rcv = port.read(20)
print "Searching : " + rcv
                print"Perform Inquiry...."
keyin = raw_input("Enter : ")
keyin2 =  keyin +'\r\n'
print "Entered Command is : "+ keyin2
port.write(keyin2)
x = 1
for x in range(0,9):
rcv = port.read(50)
print rcv
x+=1
```

```
port.flushInput()
port.flushOutput()
                print"Device address to
        which you want to Bind"
keyin = raw_input("Enter : ")
keyin2 =  keyin +'\r\n'print "Entered
Command is : "+ keyin2
                port.write(keyin2)
                rcv = port.read(50)
                print rcv
                port.flushInput()
                port.flushOutput()
                print"Paired"
time.sleep(2)
GPIO.output(25 , GPIO.LOW)
time.sleep(2)
keyin = raw_input("Want to send or
receive? ")
if keyin == "send" :
keyin = raw_input("File Name : ")
print keyin
readline = lambda :
iter(lambda:port.read(1), "\n")
port.write(open(keyin, "rb").read())
port.write("\n<<EOF>>\n")
if keyin == "receive":

port.write("<<SENDFILE>>\n")
                        readline = lambda :
iter(lambda:port.read(1), "\n")
with open("picture.jpg" , "wb") as outfile:
while True:
        line = "".join(readline())
        if line == "<<EOF>":
                break
        print >> outfile, line
        print "File Received !!!"
except:
        port.close()
```

10.5.3 Slave/Client Side code to receive file

```
import serial
port = serial.Serial("/dev/ttyAMA0" , 9600 , timeout =3.0)
port.write("<<SENDFILE>>\n")
readline = lambda : iter(lambda:port.read(1), "\n")
with open("picture.jpg" , "wb") as outfile :
        while True:
                line = "".join(readline())
                if line == "<<EOF>":
                        break
                print >> outfile, line
```

10.5.4 Output

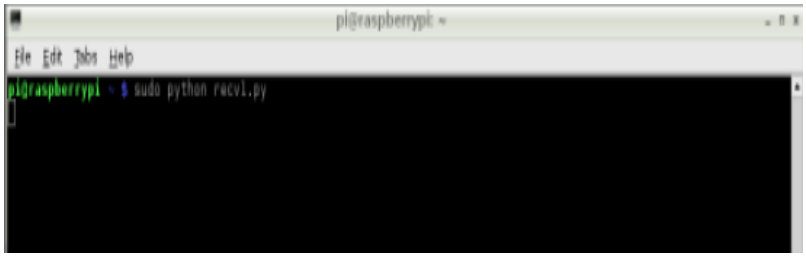


Fig.10.7: Client side code to receive file

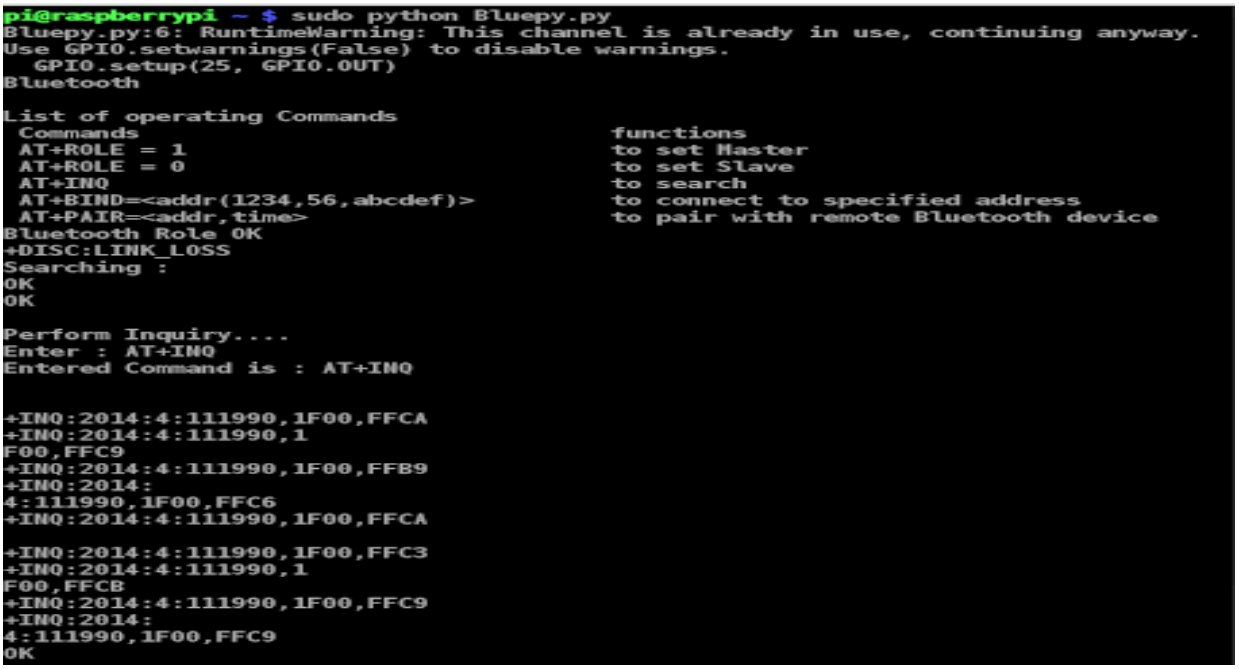


Fig.10.8: Searching nearby Bluetooth devices for sending file

Fig.10.9 : Received file by the name of Picture.jpg

10.6 Introduction : Wi-Fi

Wi-Fi is a way of connecting wirelessly to any local area network. It provides access to internet without using wires as compared to traditional wired networks like Ethernet. To use Wi-Fi, one needed to be within the range of the Wi-Fi network.

Wi-Fi is a brand owned by the Wi-Fi Alliance for wireless. It uses the IEEE 802.11 standard to communicate information between devices and/or computers. There are many different type of Wi-Fi standards available now a days that mainly differentiate between its features like distance with which devices can connect to the access points, speed at which these devices can work etc.

Most wireless networks use radio frequency band around 2.4 GHz. The 2.4 GHz band is widely used, and devices are usually cheaper.

10.6.1 Wireless USB Adapter

As Raspberry Pi is not equipped with any Wi-Fi module, this facility may be provide using either UART enable Wi-Fi modules or using USB based Wi-Fi adaptors. Any such adaptor may be used. Here TP-LINK TL-WN725N 150Mbps Wireless N Nano USB Adapter is used to connect over Wi-Fi to an 11n or other network for desired applications. Internet surfing is demonstrated here.

Fig 10.10 150Mbps Wireless N Nano USB AdapterTL-WN725N [7]

10.6.2 Features [6]

1. Seamlessly compatible with 802.11n/g/b/x/e products
2. Maximum speed of up to 150Mbps
3. Supports ad-hoc and infrastructure mode
4. Supports 64/128 WEP, WPA/WPA2, WPA-PSK/WPA2-PSK(TKIP/AES)
5. Supports IEEE 802.1x
6. Easy Wireless Configuration Utility
7. Supports Windows 8.1/8/7/XP/Vista, Mac OS X 10.7 - 10.10, Linux

10.6.3 Procedure

1. Check the version of Linux used. Use the command **uname -a** to find the version of Linux.
   e.g. Here version of Linux is 4.1.6-v7
2. Connect wifi dongle -TP-LINK TL-WN725N V2 or similar wifi dongles using the 8188eu driver module for systems using the **Raspbian** image.
3. With the wifi dongle connected to Pi use command **lsusb** to show a list of USB devices connected to Pi. USB ID is needed to check driver support.

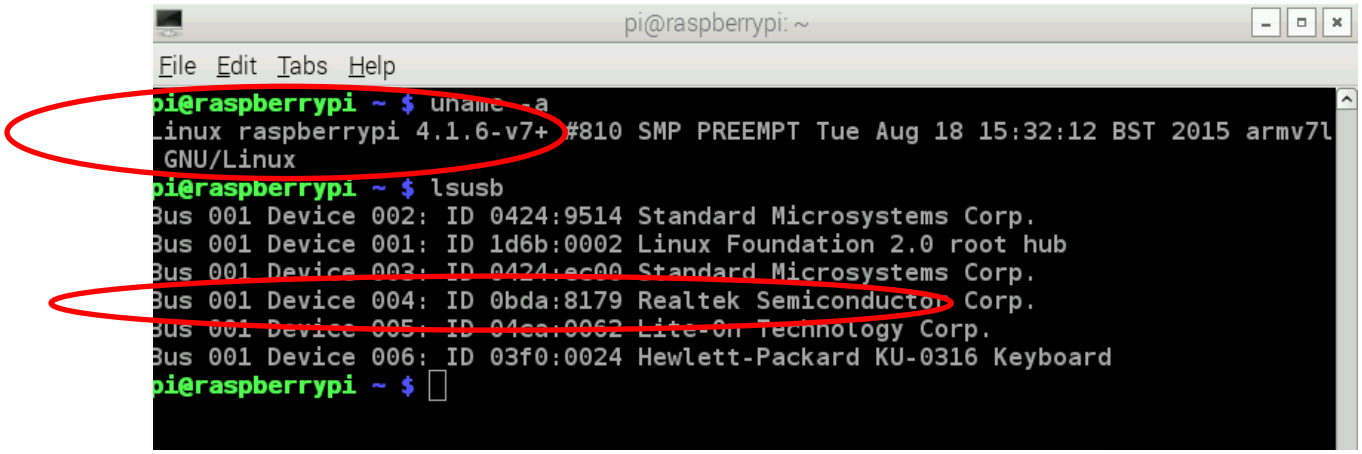


Fig: 10.11 Checking Linux version and USB driver module

4. According to USB ID, Get Realtex drivers for Pi 2 using command
   a. **$ sudo wget https://dl.dropboxusercontent.com/u/80256631/8188eu-2015yyzz.tar.gz**

   Replace **yyzz with version of kernel as seen using uname –a command. E.g Here it is 0818 Then write
   b. **$ sudo wget https://dl.dropboxusercontent.com/u/80256631/8188eu-20150818.tar.gz**
5. Unzip this downloaded driver using command
   $ sudo tar xzf 8188eu-2015yyzz.tar.gz
6. Install this driver using command
   ./install.sh
7. Reboot the system

Installation will install the driver file and copy the file **8188eu.conf** to directory **/etc/modprobe.d**. The 8188eu.conf file will disable power management for the 8188eu driver and disable the inbuilt

driver version r8188eu. Once the driver is loaded, configuration of the network set up is needed so the Pi will connect to the Wi-Fi network [8].

10.6.4 Configuring Wi-Fi network for raspbian using wpa_supplicant

1. Type $ sudo nano /etc/network/interfaces

This will open default **/etc/network/interfaces** file as supplied in the raspbian image. Provide static IP and edit file as given in fig 10.12.

** Do not edit if IP address supports dhcp protocol.

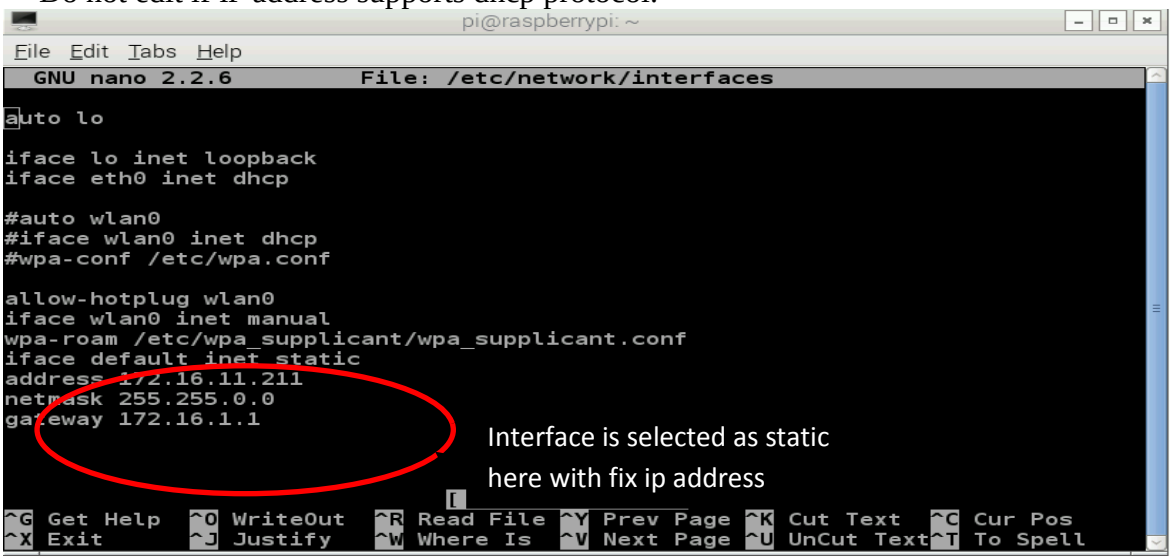


Fig: 10.12 Interface settings for static IP address

2. Open and edit wpa_supplicant.conf file and add the network section. Use command
$ **sudo nano /etc/wpa_supplicant/wpa_supplicant.conf** after changes exit the editor and save the file using keys cntl-X, Y, Enter.

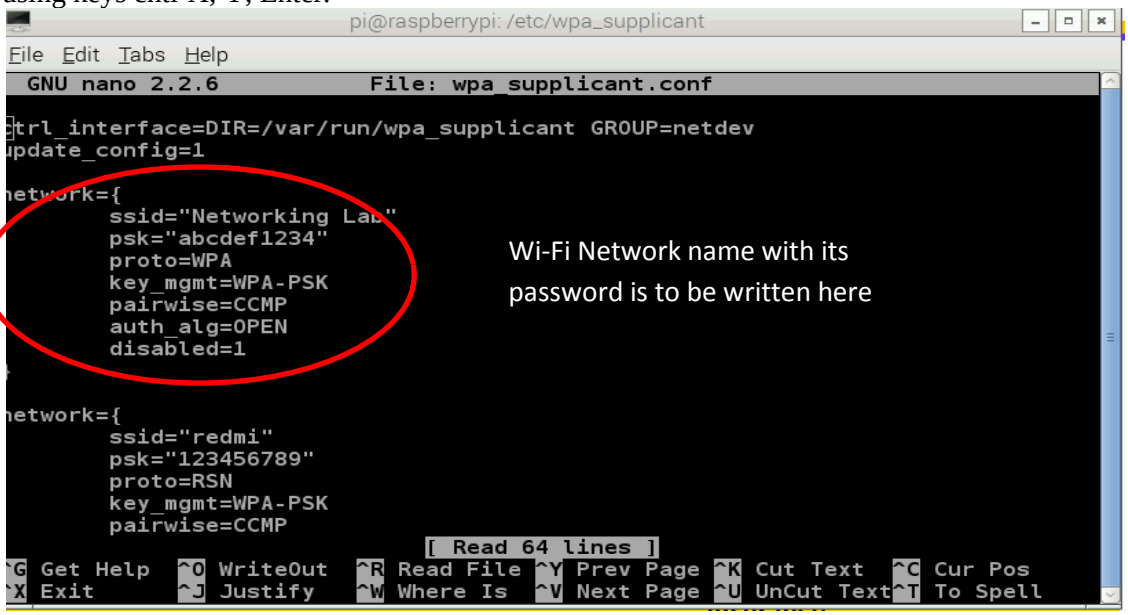


Fig: 10.13 Changes in wpa_supplicant.conf file for adding network details

3. At Raspberry Pi2 start Menu Open Menu>Preference>Wi-Fi Configuration

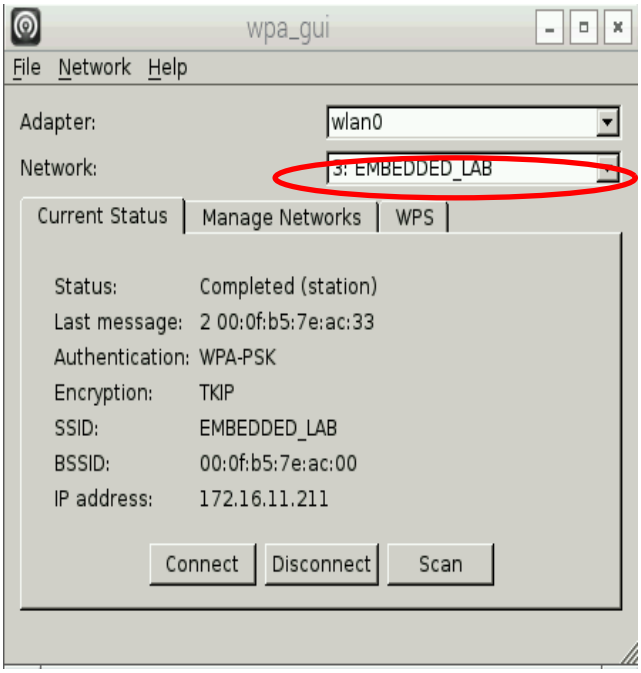


Fig 10.14 a: Configuring Wi-Fi connection

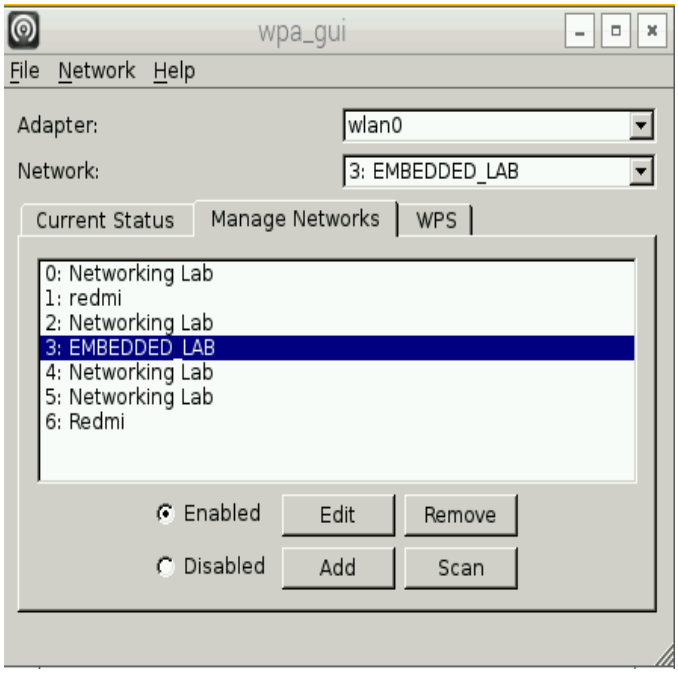


Fig 10.14 b: Select available Network

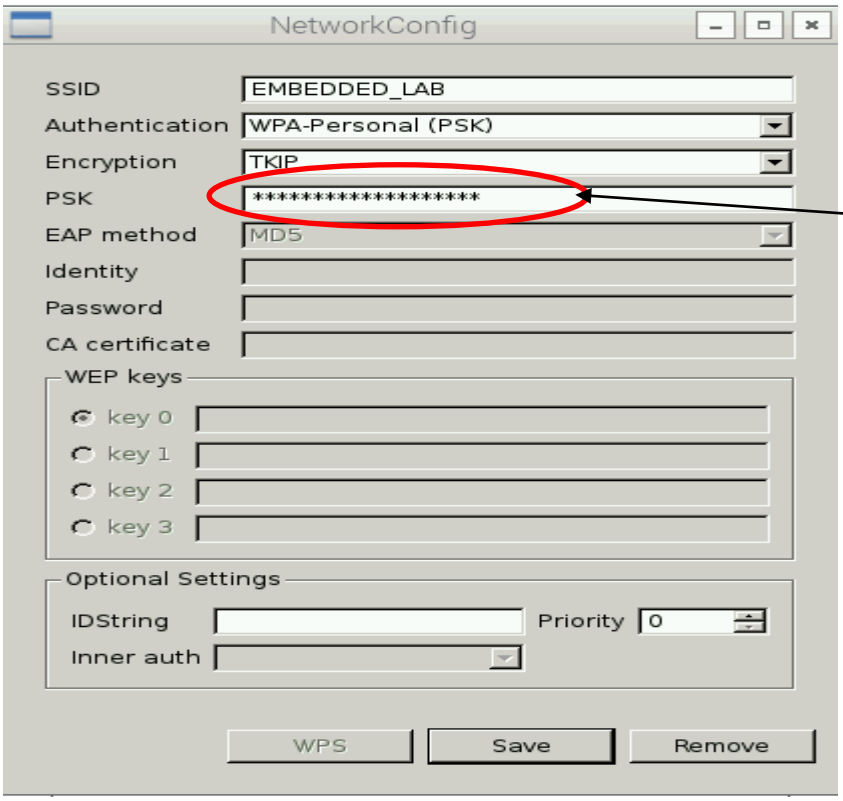


Fig 10.14 c: Enter available Network password

Wi-Fi interfacing is completed and we can access the website. The www.igdtuw.ac.in is accesses as shown in Fig 10.15.

4.Use Menu>Internet>**Iceweasel or Midori** for browsing.

10.6.5 Output

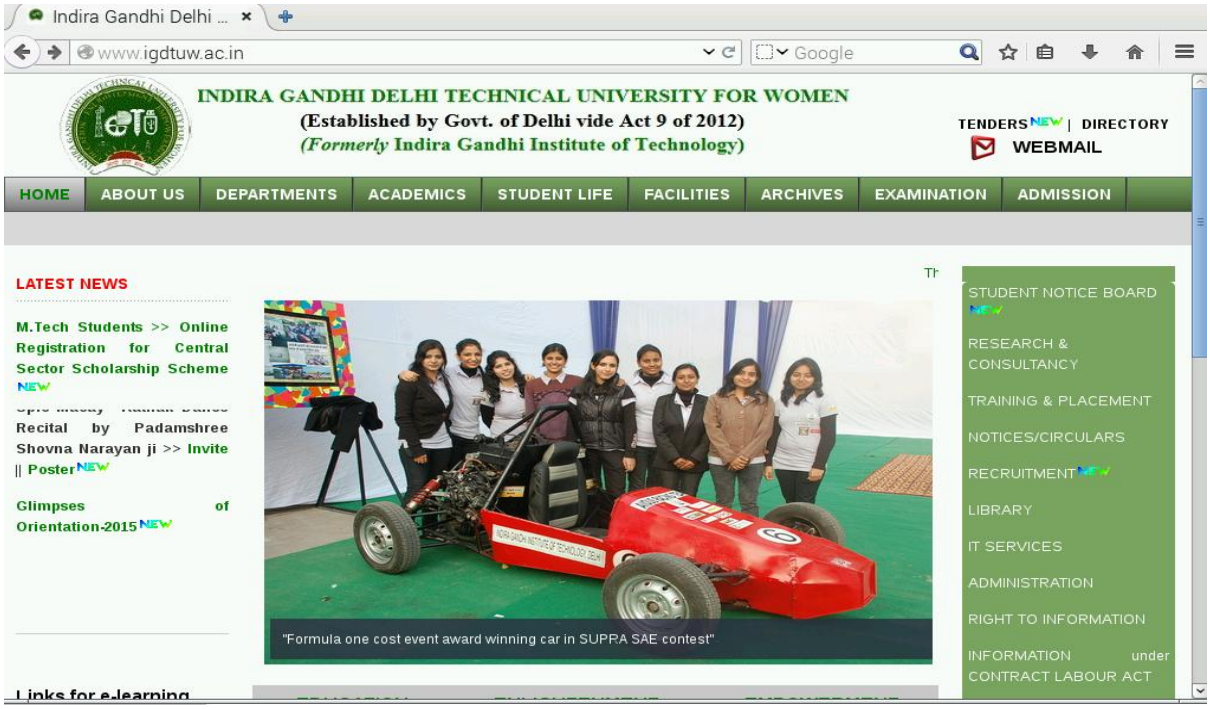


Fig 10.15 IGDTUW website is browsed using Wi-Fi

Exercise:

1. Write a program to execute more AT commands and produce their outputs.
2. Write a program to connect more Bluetooth modules and form piconet.
3. Write python code for combining above all functionalities in one go.

References:

[1] http://elinux.org/RPi_Serial_Connection"
[2]http://www.raspberry-projects.com/pi/programming-in-c/uart-serial-port/using-the-uart
[3]http://www.tec.reutlingen-university.de/uploads/media/DatenblattHC-05_BT-Modul.pdf
[4]http://eskimon.fr/wp-content/uploads/2014/10/commandes_AT_HC05.pdf
[5]Bluetooth SPP Manager, "https://play.google.com/store/apps/details?id=a t.rtcmanage r &hl=en
[6] http://www.tp-link.com/res/down/doc/TL-WN725N_V2_UG.pdf
[7] http://www.tp-link.com/lk/products/details/cat-11_TL-WN725N.html
[8] http://www.fars-robotics.net/

Chapter- 11
Sensor Interfacing and Programming

11.1 Introduction

A sensor is a transducer which converts physical parameters into electrical parameters. For example, a thermocouple converts temperature to an output voltage, LDR converts light into voltage, Piezo-electric transducer converts pressure into voltage. This chapter shows the interfacing of various sensors with the Raspberry Pi.

11.2 DHT 11 SENSOR

DHT11 Sensor senses temperature & humidity together and gives calibrated digital signal as output. By using the exclusive digital-signal-acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability. This sensor includes a resistive-type humidity measurement component and an NTC temperature measurement component, and connects to a high-performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost-effectiveness. The features of DHT11 includes:

1. Relative humidity and temperature measurement
2. Calibrated digital signal
3. Outstanding long-term stability
4. Long transmission distance
5. Low power consumption
6. Resistive-type humidity and temperature sensor
7. Body size 15.5mm x 12mm x 5.5mm

11.2.1 Interfacing DHT 11 sensor with Raspberry pi

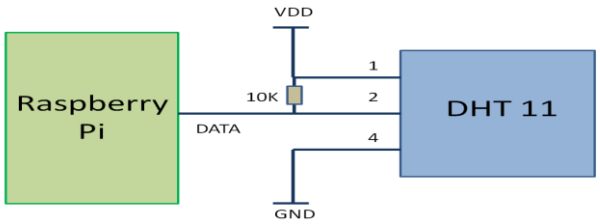


Figure 11.1: circuit diagram

It is easy to connect DHT11 to the Raspberry Pi2. The code can use any GPIO pin, but GPIO 4 is used in diagrams and codes.

11.2.2 Programming of Raspberry Pi 2 for DHT 11

For running the DHT11 python code on raspberry pi, we need to install the DHT driver from github.

Downloading the Code from Github

The easiest way to get the code onto Pi is to hook up an Ethernet cable, and clone it directly using 'git', which is installed by default. Simply run the following commands from an appropriate location (ex. "/home/pi")[1]:

- **Download Adafruit_Python_DHT-master.zip from github**

  $ cd Adafruit-Raspberry-Pi-Python-Code
  $ cd Adafruit_DHT_Driver_Python
- **Download bcm2835 library from the given link.**

  http://www.airspayce.com/mikem/bcm2835/
  Extract it in /home/pi location.
- **Configure the bcm2835-1.42 by using given commands**

  $ cd bcm2835-1.42
  $ ./ configure
  $ make
  $ make install
- **Download setuptools-4.0.1**

  $ cd setuptools-4.0.1
  $ sudo python setup.py build
- **Update the OS and install python-dev**

  $ sudo apt-get update
  $ sudo apt-get install build-essential python-dev
  $ cd Adafruit_python_DHT-master
  $ sudo python setup.py install

Program:

```
import dhtreader
import time
type = 11
pin = 04
dhtreader.init()
while True:
        try:
                temp, hum=dhtreader.read(type, pin)
        except TypeError:
                if temp and hum:
                        print dhtreader.read(type, pin)
                        print "Temp=",temp,"*c"
                        print "Humidity=", hum,"%","\n"
                else:
                        print "No data"
                time.sleep(2)
```

11.2.3 DHT11 Readings

A.) at room temperature

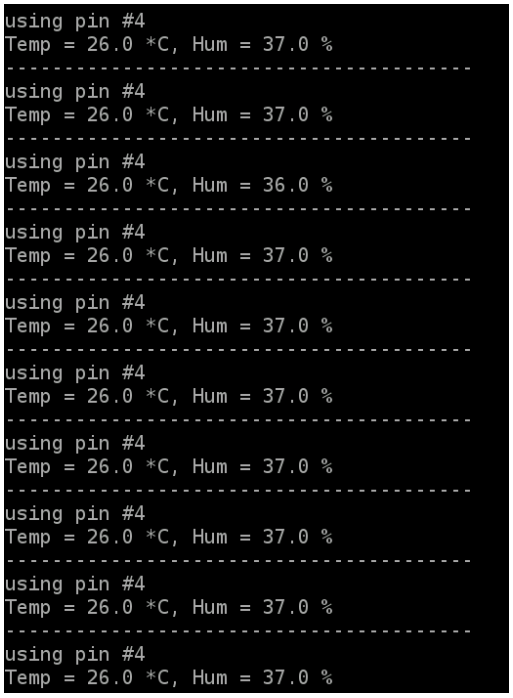


Fig 11.2: Output readings at room temperature

B.) after increasing the temperature (with the help of candle/matchstick)

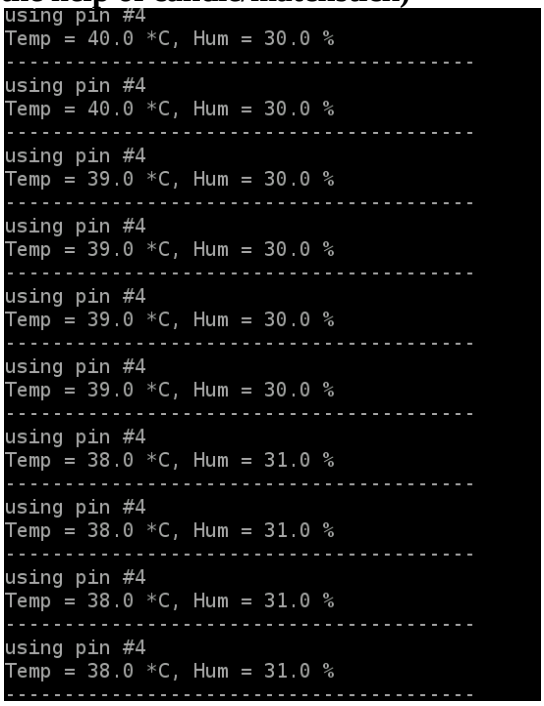


Fig 11.3: Output readings after increasing temperature

11.3 Light Sensor

A light sensor is a device that is used to detect light. There are many different types of light sensors each of which works in different way. A photo resistor is a small sensor that changes its resistance when light shines on it. These are used in many consumer products to determine the intensity of light. A charged coupled device (CCD) transports electrically charged signals, and is used as a light sensor in digital cameras and night-vision devices. Photomultipliers detect light and multiply it.

TSL2561: The TSL2561 luminosity sensor is an advanced digital light sensor ideal for use in a wide range of light situations. Compared to low cost CdS cells, this sensor is more precise allowing for exact Lux calculations and can be configured for different gain/timing ranges to detect light ranges from up to 0.1 - 40,000+ Lux on the fly. The best part of this sensor is that it contains **both infrared and full spectrum diodes**. This means one can separately measure infrared, full-spectrum or human-visible light.

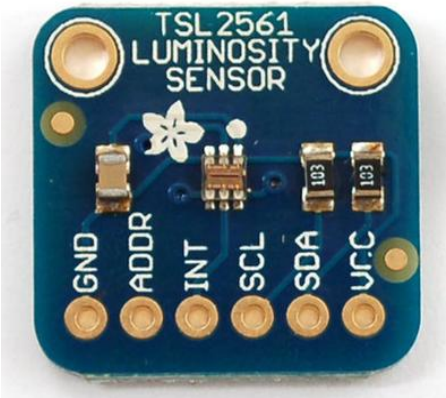


Fig 11.4 Light Sensor TSl2561 module

11.3.1 Interfacing of TSL2561 Sensor with Raspberry Pi

1. Connect the **VCC** pin to a **3.3V** power source. The sensor cannot be used with anything **higher than 3.3V so don't use a 5V supply.**
2. Connect **GND** to the ground pin.
3. Connect the **I2CSCL clock** pin to your I2C clock pin.
4. Connect the **I2C SDA data** pin to your I2C data pin as shown in Fig 11.5

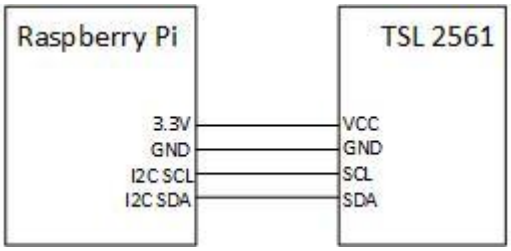


Fig. 11.5 Interfacing of TSL2561 with Raspberry Pi

Note: No need to connect the **ADDR** (I2C address change) or **INT** (interrupt output) pins.

The **ADDR** pin can be used if one has an I2C address conflict, to change the address.

1. Connect it to ground to set the address to **0x29**,
2. Connect it to 3.3V (VCC) to se t the address to **0x49**
3. Or, leave it floating (unconnected) to use address **0x39**.

The **INT** pin is an output from the sensor, when one has the sensor configured to signal when the light level has changed. To use INT pin, use 10K-100K pull-ups from **INT** to 3.3V (VCC)

11.3.2 Programming of Raspberry Pi for TSL 2561

I2C Communication on raspberry pi: I²C (Inter-Integrated Circuit), pronounced *I-squared-C*, is a multi-master, multi-slave, single-ended, serial computer bus invented by NXP Semiconductors, used for attaching low-speed peripherals to computer motherboards and embedded systems.

Step 1: Enable I2C

On the Pi, I2C is disabled by default and it need to be enabled using following command

sudo nano /etc/modprobe.d/raspi-blacklist.conf

In this file, there is a comment, and two lines. Add a hash before the I2C line, to comment it out.

Original:

 # blacklist spi and i2c by default (many users don't need them)
blacklist spi-bcm2708
blacklist i2c-bcm2708

This means the i2c is disabled and users can't use it. For enabling i2c, use '#' in front of i2c line as shown in screenshot below

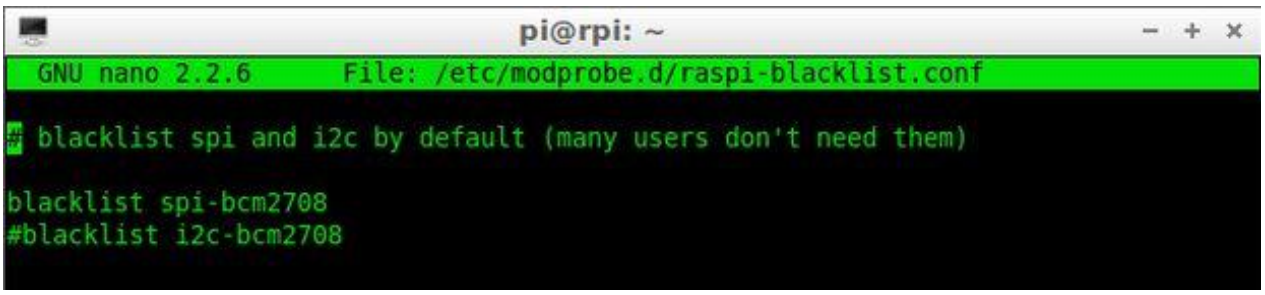


Fig 11.6: Enable I2C

This means the i2c is enabled and users can use it.

Step 2: Enable kernel I2C Module

The next thing to do is add the I2C module to the kernel. Run the command

sudo nano /etc/modules

One should see the following file:

 # /etc/modules: kernel modules to load at boot time.
#
 # This file contains the names of kernel modules that should be loaded
 # at boot time, one per line. Lines beginning with "#" are ignored.
 # Parameters can be specified after the module name.

snd-bcm2835

This should have the line **i2c-dev** added to the end for adding i2c module. Final file:

 # /etc/modules: kernel modules to load at boot time.
#
 # This file contains the names of kernel modules that should be loaded
 # at boot time, one per line. Lines beginning with "#" are ignored.
 # Parameters can be specified after the module name.

snd-bcm2835
i2c-dev

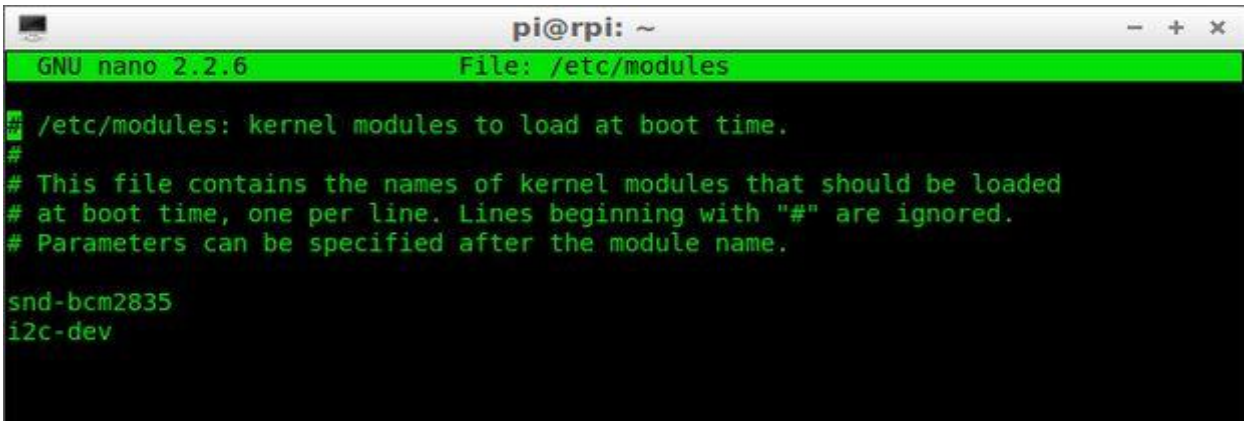


Fig 11.7: Enable kernel I2C module

Step 3: Install Necessary Packages

There are a few packages that will need installing to use I2C. The first command to run is **sudo apt-get install i2c-tools**

If this fails, try running **sudo apt-get update**

The other package needed can be installed by running **sudo apt-get install python-smbus**

To configure the software, one has to add the Pi user to the I2C access group, by running the command **sudo adduser pi i2c**

Now run **sudo reboot** to reboot, and test the new software.
To test the software, run the command **sudo i2cdetect –y,** to see if there is anything connected.

In this setup, it returned following output as shown in figure 11.9, because TSL2561 is connected:
For TSL2561connection, connect the SDA and SCL lines to the Pi SDA and SCL, and one is ready to roll.

When TSL2561 sensor connected to I2C then address of sensor will be shown as:

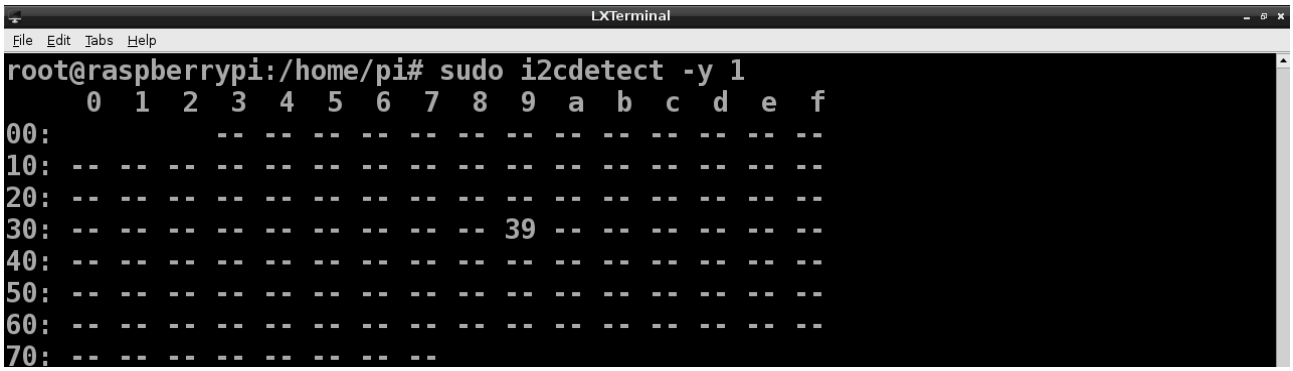


Fig 11.8: Detection of I2C device

Programming for TSL2561:

The Adafruit _I2C is in the adafruit Raspberry Pi Python Code GitHub repository. Check out a copy of it, and change into Adafruit directory:

$ git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code
$ cd https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code/Adafruit_I2C.

```
#!/usr/bin/python

import sys
import smbus
import time
from time import sleep
from Adafruit_I2C import Adafruit_I2C
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setup(25,GPIO.OUT)
GPIO.setwarnings(False)

class Luxmeter:
        i2c = None
        gain = 0
        def __init__(self, address=0x39, debug=0, pause=0.8):
                self.i2c = Adafruit_I2C(address)
                self.address = address
                self.pause = pause
                self.debug = debug
                self.gain = 0
                self.i2c.write8(0x80, 0x03)
def setGain(self,gain=1):
                """set the gain"""
```

```
                if(gain != self.gain):
                        if(gain==1):
                                self.i2c.write8(0x81,0x02)
                                if (self.debug):
                                        print"Setting low gain"
                        else:
                                self.i2c.write8(0x81, 0x12)
                                if(self.debug):
                                        print"Setting high gain"
                        self.gain=gain;
                        time.sleep(self.pause)
def readWord(self, reg):
                        """Reads a word from I2C device"""
                        try:
                                wordval = self.i2c.readU16(reg)
                                newval = self.i2c.reverseByteOrder(wordval)
                                if (self.debug):
print("I2C: Device 0x%02X returned 0x%04X from reg 0x%02X"%(self.address, wordval &
0xFFFF, reg))
                                return newval
                        except IOError:
                                print("Error accessing 0x%02X: Check your I2C address" %
self.address)
                                return -1

        def readFull(self, reg=0x8C):
                        """Reads visible+IR diode from the I2C device"""

                        return self.readWord(reg);
                def readIR(self, reg=0x8E):
                        """Reads IR only diode from the I2C device"""
                        return self.readWord(reg);
def getLux(self, gain = 0):
"""Grabs a lux reading either with autoranging (gain=0) or with a specified gain (1, 16)"""
                        if (gain == 1 or gain == 16):
                                self.setGain(gain)
                             ambient = self.readFull()
                          IR = self.readIR()
                        elif (gain==0):
                                self.setGain(16)
                                ambient = self.readFull()
                                if (ambient < 65535):

IR = self.readIR()
                                if (ambient >= 65535 or IR >= 65535):
                                        self.setGain(1)
```

```
                                ambient = self.readFull()
                                IR = self.readIR()

                        if(self.gain==1):
                                ambient *= 16
                                IR *= 16
                        if (IR <= 0):
                    print"LIGHT ON"
                                GPIO.output(25,GPIO.HIGH)
                        elif (IR >= 16):
                                GPIO.output(25, GPIO.LOW)
                        ratio = (IR / float(ambient))
                        if (self.debug):
                                print "IR Result", IR
                     print "Ambient Result", ambient

                        if ((ratio >= 0) & (ratio <= 0.52)):
                                lux = (0.0315 * ambient) - (0.0593 * ambient * (ratio**1.4))
                        elif (ratio <= 0.65):
                                lux = (0.0229 * ambient) - (0.0291 * IR)
                        elif (ratio <= 0.80):
                                lux = (0.0157 * ambient) - (0.018 * IR)
                        elif (ratio <= 1.3):
                                lux = (0.00338 * ambient) - (0.0026 * IR)
                      elif (ratio > 1.3):
                                lux = 0
                     return lux

while True:
        oLuxmeter = Luxmeter()
        print "Light intensity in term of LUX ",  oLuxmeter.getLux(1)
        print "================================================="
        sleep(.1)
```

11.3.3 Output of this experiment is shown in Fig 11.9

Fig: 11.9 Light Intensity output using light sensor TSl2561

11.4 Accelerometer

The ADXL345 is a small, thin, low power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16g. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I2C digital interface. The ADXL345 is well suited for mobile device applications. It measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (4 mg/LSB) enables measurement of inclination changes less than 1.0°. Several special sensing functions are provided. Activity and inactivity sensing detect the presence or lack of motion and if the acceleration on any axis exceeds a user-set level. Tap sensing detects single and double taps. Free-fall sensing detects if the device is falling. These functions can be mapped to one of two interrupt output pins. An integrated, patent pending 32-level first in, first out (FIFO) buffer can be used to store data to minimize host processor intervention

FEATURES:

- Ultralow power: as low as 23 µA in measurement mode and 0.1 µA in standby mode at VS = 2.5V
- Power consumption scales automatically with bandwidth
- User-selectable resolution
    - Fixed 10-bit resolution Full resolution, where resolution increases with g range, up to
    - 13-bit resolution at ±16 g (maintaining 4 mg/LSB scale factor in all g ranges)
- Supply voltage range: 2.0 V to 3.6 V
- I/O voltage range: 1.7 V to Vs
- SPI (3- and 4-wire) and I2C digital interfaces
- Flexible interrupt modes mappable to either interrupt pin
- Measurement ranges selectable via serial command
- Bandwidth selectable via serial command
- Wide temperature range (−40°C to +85°C)
- 10,000 g shock survival
- Pb free/RoHS compliant
- Small and thin: 3 mm × 5 mm × 1 mm LGA package

APPLICATIONS:

- Handsets
- Medical instrumentation
- Gaming and pointing devices
- Industrial instrumentation
- Personal navigation devices
- Hard disk drive (HDD) protection

11.4.1 Interfacing of ADXL345 with Raspberry pi 2

The sensor has a digital (I2C) interface. One can select one of three addresses so one can have up to three sensors on one board - each with a different i2c address. The built in ADC means it can be directly used with any microcontroller, even if it doesn't have analog inputs. The current draw is extremely low, so it's great for low power data-logging systems about 0.5mA when actively sensing, and less than 15 µA when in power down mode.

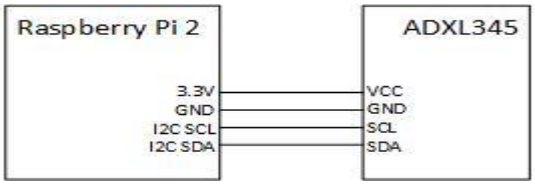


Fig 11.10: Accelerometer Sensor Interfacing

- Connect the **VCC** pin to a **3.3V** power source. The sensor cannot be used with anything higher than 3.3V so don't use a 5V supply!
- Connect **GND** to the ground pin.
- Connect the **i2c SCL clock** pin to your i2c clock pin.
- Connect the **i2c SDA data** pin to your i2c data pin

11.4.2 Programming of Raspberry Pi 2 with ADXL345

Download the ADXL345 pimoroni python library from github:

git clone https://github.com/pimoroni/adxl345-python

The adxl345-python project from pimoroni contains a python module for reading data from the ADXL345 perhaps not unsurprisingly called "adxl345.py", inside there is a class called "ADXL345" which is how you to interact with the accelerometer.
The program below imports the module, instantiates an ADXL345 object and reads values from the accelerometer as g-forces.

```
#import the adxl345 module
import adxl345
#create ADXL345 object
accel = adxl345.ADXL345()
#get axes as g
axes = accel.getAxes(True)
# to get axes as ms^2 use
#axes = accel.getAxes(False)
#put the axes into variables
x = axes['x']
y = axes['y']
z = axes['z']
#print axes
print x
print y
print z
```

11.4.3 Output

```
pi@devpi ~/dev/adxl345-python $ sudo python example.py
ADXL345 on address 0x53:
   x = 0.056G
   y = 0.088G
   z = 0.872G
pi@devpi ~/dev/adxl345-python $
```

Fig 11.11 Output readings for ADXL345

11.5 Ultra Sonic Sensor

HC-SR04: Ultrasonic ranging module is an active sensor. It has four pins as shown in the below diagram. When Trig pin is set to HIGH level for at least 10us, the module automatically sends eight 40 kHz sound pulses and detect whether there is pulse of back signal. If there is any pulse then it makes Echo pin HIGH for time period equal to the time between sending and receiving the pulses.

Distance = (HIGH level time of Echo pin/2) * velocity of sound (340M/S)

Specifications [2]:

Working Voltage, Current	DC 5 V, 15 mA
Working Frequency	40 KHz
Range	2cm-4m
Measuring Angle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension (mm)	45*20*15

11.5.1 Interfacing of Ultrasonic Sensor with Raspberry Pi 2

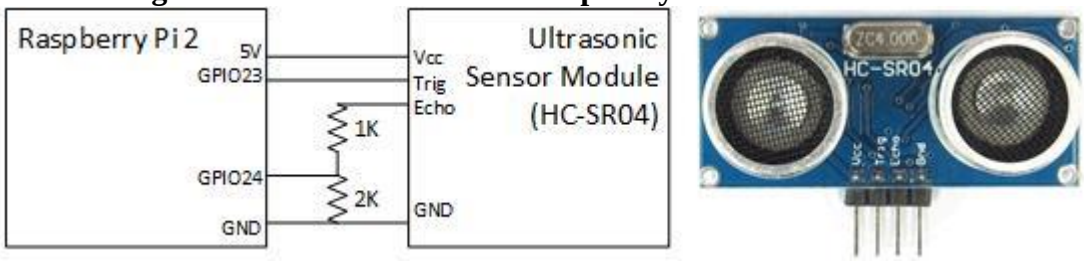


Fig 11.12 Ultrasonic Sensor Interfacing

Make the connections as given below

Ultrasonic Sensor	**Raspberry Pi 2**
Vcc	+5v
GND	GND
Trig	GPIO 23
Echo	GPIO 24

11.5.2 Programming of Raspberry Pi 2 for Ultrasonic sensor

1. Install the GPIO package for Python by following command in terminal:
   $ sudo apt-get install python-dev python-rpi.gpio

2. Write the following Python program in any text editor and save it as Ultrasonic.py

```
import RPi.GPIO as GPIO                 #Import GPIO library
import time                             #Import time library
GPIO.setmode(GPIO.BCM)                  #Set GPIO pin numbering

TRIG = 23                               #Associate pin 23 to TRIG
ECHO = 24                               #Associate pin 24 to ECHO

print "Distance measurement in progress"
GPIO.setup(TRIG,GPIO.OUT)               #Set pin as GPIO out
GPIO.setup(ECHO,GPIO.IN)                #Set pin as GPIO in

while True:
 GPIO.output(TRIG, False)               #Set TRIG as LOW
 print "Waitng For Sensor To Settle"
 time.sleep(2)                          #Delay of 2 seconds

 GPIO.output(TRIG, True)                #Set TRIG as HIGH
 time.sleep(0.00001)                    #Delay of 0.00001 seconds
 GPIO.output(TRIG, False)               #Set TRIG as LOW
 while GPIO.input(ECHO)==0:             #Check whether the ECHO is LOW
  pulse_start = time.time()             #Saves the last known time of LOW pulse
 while GPIO.input(ECHO)==1:             #Check whether the ECHO is HIGH
  pulse_end = time.time()               #Saves the last known time of HIGH pulse

 pulse_duration = pulse_end - pulse_start #Get pulse duration to a variable

 distance = pulse_duration * 17150      #Multiply pulse duration by 17150 to get distance
 distance = round(distance, 2)          #Round to two decimal points

 if distance > 2 and distance < 400:    #Check whether the distance is within range
  print "Distance:",distance - 0.5,"cm"  #Print distance with 0.5 cm calibration
 else:
  print "Out Of Range"                  #display out of range
```

11.5.3 Output

3. Run the program in the terminal by following command
   $ sudo python Ultrasonic.py

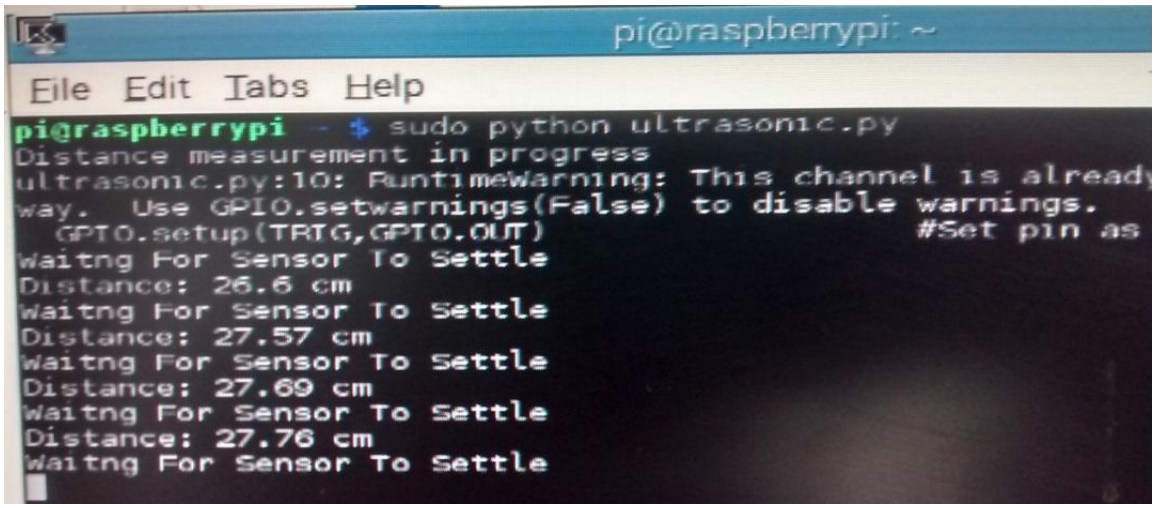


Fig 11.13 Ultrasonic sensor

11.6 Sound Sensor

This sensor listens to incoming sound through MIC. When it detects sound, it makes output HIGH for a preset amount of time. The output duration of the high signal can be adjusted by potentiometer. During silence there is no output from module. This module is useful in making projects where sound activation features are required like Security Camera which takes pictures upon sound activity. This sensor module is not intelligent to separate different sounds, this is a low cost alternative for sound controlling applications [3].

11.6.1 Interfacing of Sound Detector with Raspberry Pi 2

Fig 11.14 Sound Detector Interfacing

Make the connections as given below

Sound Detector	Raspberry Pi 2
Vcc	+5v
GND	GND
Output	GPIO 4

11.6.2 Programming of Raspberry Pi 2 for Sound sensor

4. Install the GPIO package for Python by following command in terminal:
   $ sudo apt-get install python-dev python-rpi.gpio

5. Write the following Python program in any text editor and save it as Sound.py

```
Import Rpi.GPIO as GPIO          #Import GPIO library
import time                      #Import time library
GPIO.setmode(GPIO.BCM)           #Set GPIO pin numbering
sound = 4                            #Associate pin 26 to pir
GPIO.setup(sound, GPIO.IN)           #Set pin as GPIO in
while True:
   if GPIO.input(sound):              #Check whether pir is HIGH
      print "Sound Detected"
   else
      print "No Sound"
   time.sleep(2)                  #Delay to avoid multiple detection
```

6. Run the program in the terminal by following command
   $ sudo python Sound.py

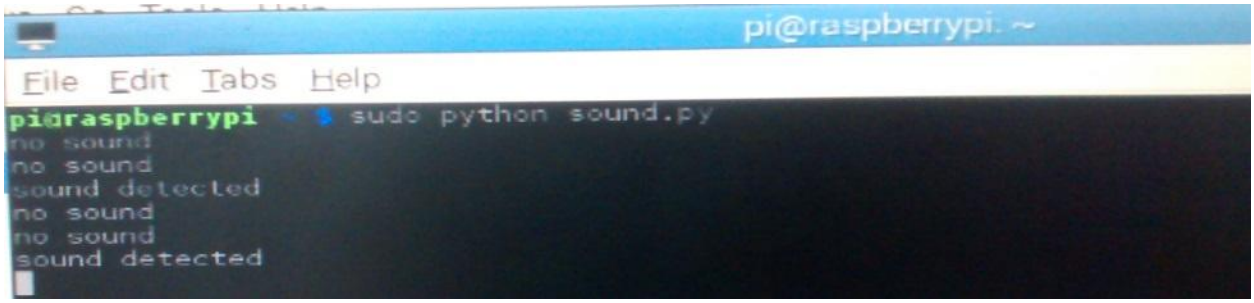


Fig 11.15 Sound sensor

11.7 LDR Sensor

A light dependent resistor (LDR) is a light-controlled variable resistor. The resistance of a photo-resistor decreases with increasing incident light intensity. In other words, it exhibits photoconductivity [4].

LDR module can be applied in and light and dark activated switching circuits. LDR module consists of an LDR-Resister voltage divider, potentiometer and a comparator as shown in circuit below. The negative input of the comparator is connected to potentiometer which is used as threshold voltage. Normally LDR resistance is high and voltage at positive input of the comparator is lower than voltage at negative input. So comparator output is 0 volts (LOW). When light falls on LDR, then resistance of the LDR decreases, voltage at positive input of the comparator is higher than voltage at negative input. So comparator output is 5 volts (HIGH). A switch is used to exchange the comparator inputs, if reverse output is desired.

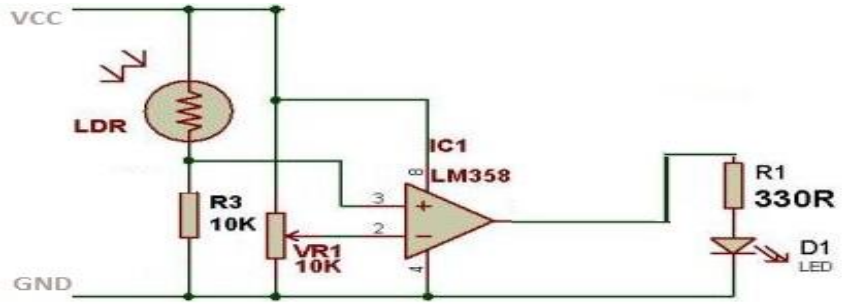


Fig 11.16 LDR Module [5]

11.7.1 Interfacing of LDR Sensor with Raspberry Pi 2

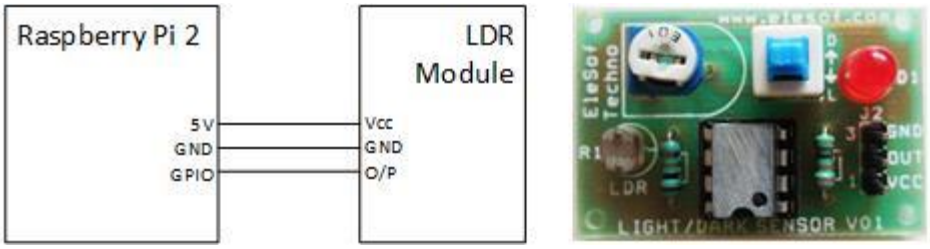


Fig 11.17 LDR Sensor Interfacing

Make the connections as given below

IR Sensor	Raspberry Pi 2
Vcc	+3.3v
GND	GND
Output	GPIO 4

11.7.2 Programming of Raspberry Pi 2 for LDR sensor

7. Install the GPIO package for Python by following command in terminal:
   $ sudo apt-get install python-dev python-rpi.gpio
8. Write the following Python program in any text editor and save it as LDR.py

```
Import Rpi.GPIO as GPIO          #Import GPIO library
import time                      #Import time library
GPIO.setmode(GPIO.BCM)           #Set GPIO pin numbering
LDR = 4                            #Associate pin 26 to pir
GPIO.setup(LDR, GPIO.IN)           #Set pin as GPIO in
while True:
   if GPIO.input(LDR):               #Check whether pir is HIGH
      print "Light Detected"
   else
      print "No Light"
   time.sleep(2)                  #Delay to avoid multiple detection
```

9. Run the program in the terminal by following command
   $ sudo python LDR.py

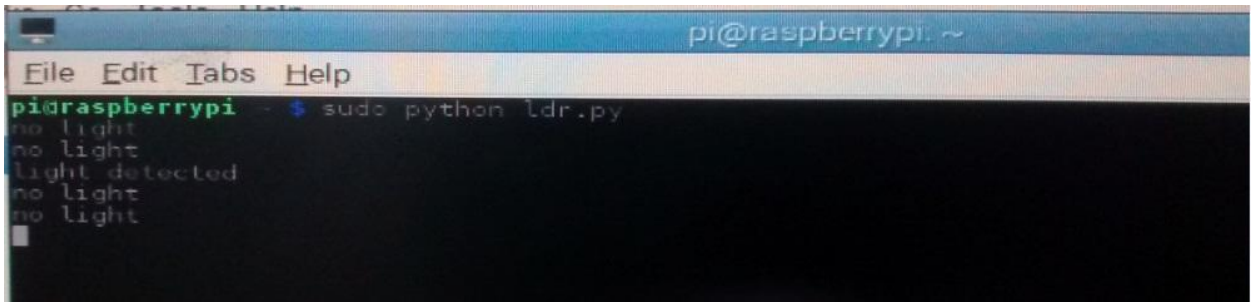


Fig 11.18 LDR Sensor

11.8 IR Sensor

IR obstacle sensor module consists of an IR LED, IR photodiode, potentiometer and a comparator. The negative input of the comparator is connected to potentiometer which is used as threshold voltage. Reverse-biased photodiode resistance is high and voltage at positive input of the comparator is lower than voltage at negative input. So comparator output is 0 volts (LOW). When IR waves emitted from IR LED are reflected from an obstacle to the photodiode, then resistance of the photodiode decreases, voltage at positive input of the comparator is higher than voltage at negative input. So comparator output is 5 volts (HIGH).

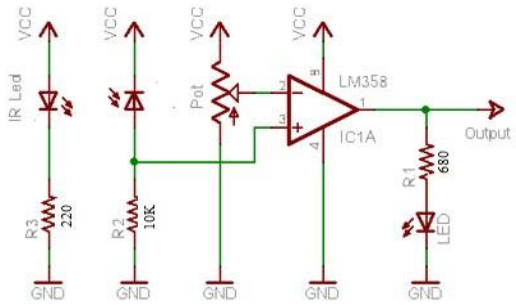


Fig 11.19 IR Sensor Module [6]

11.8.1 Interfacing of IR sensor with Raspberry Pi 2

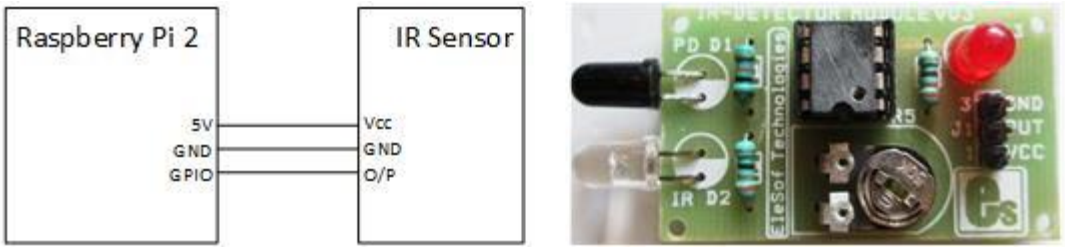


Fig 11.20 IR Sensor Interfacing

Make the connections as given below

IR Sensor	Raspberry Pi 2
Vcc	+3.3v
GND	GND
Output	GPIO 4

11.8.2 Programming of Raspberry Pi 2 for IR sensor

10. Install the GPIO package for Python by following command in terminal:
    $ sudo apt-get install python-dev python-rpi.gpio
11. Write the following Python program in any text editor and save it as IR.py

```
Import Rpi.GPIO as GPIO          #Import GPIO library
import time                      #Import time library
GPIO.setmode(GPIO.BCM)           #Set GPIO pin numbering
IR = 4                           #Associate pin 26 to pir
GPIO.setup(IR, GPIO.IN)          #Set pin as GPIO in
while True:
   if GPIO.input(IR):            #Check whether pir is HIGH
      print "Obstacle Detected"
   else
      print "No Obstacle"
   time.sleep(2)                 #Delay to avoid multiple detection
```

12. Run the program in the terminal by following command
    $ sudo python IR.py

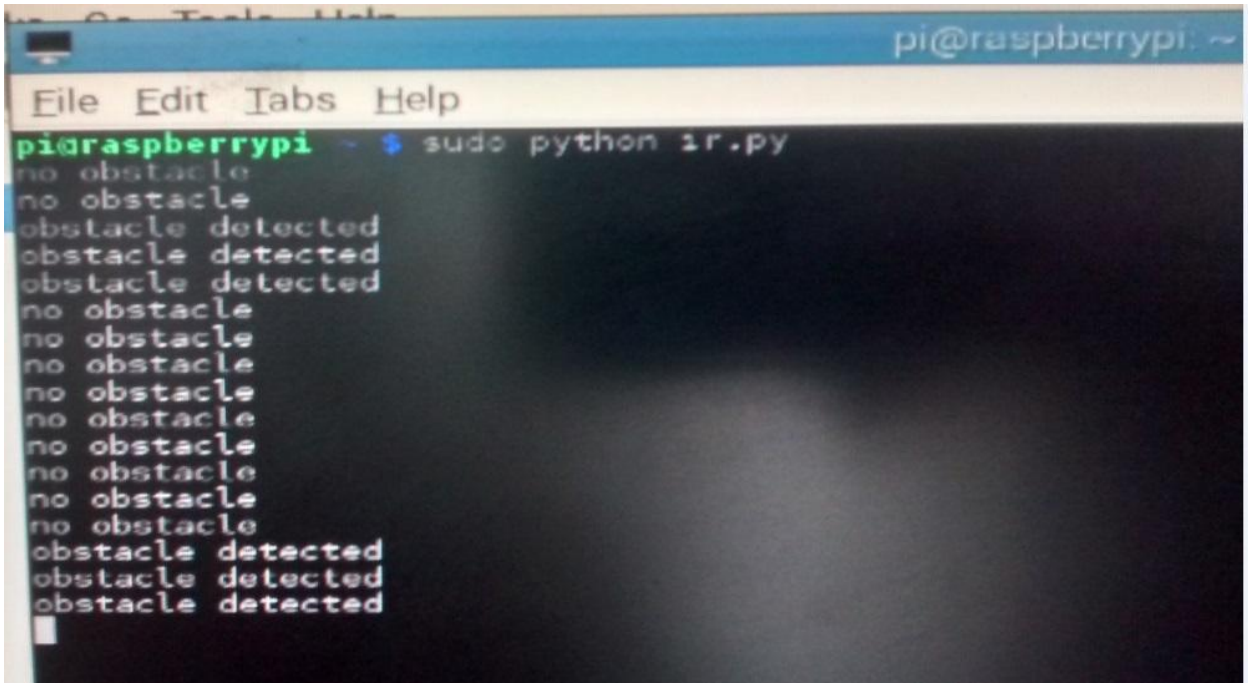
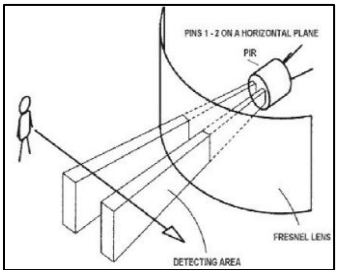


Fig 11.21 IR Sensor

11.9 PIR Sensor

A passive infrared sensor (PIR sensor) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in motion detectors [7]. PIRs are basically made of a pyro electric sensor, which can detect levels of IR radiations. Everything emits some low level radiation, and the hotter things emits more radiation. The sensor in a motion detector is split in two halves. If one half sees more or less IR radiation than the other, the output will swing high or low. The PIR sensor itself has two slots in it, each slot is made of a special material that is sensitive to IR. When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the room or walls or outdoors. When a warm body like a human or animal passes in front of it, it first intercepts one half of the PIR sensor, which causes a positive differential change between the two halves. When the warm body second half of PIR, the sensor generates a negative differential change. These change pulses are detected and output is shown in below diagrams.



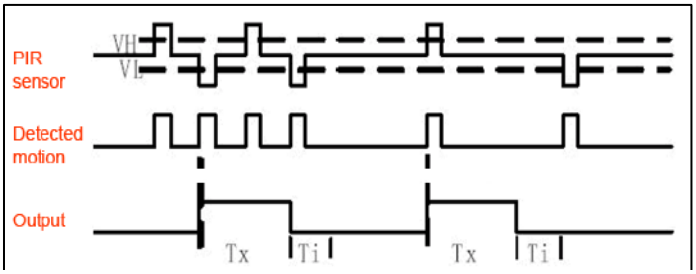


Fig 11.22 (a) PIR Sensor [8] (b) PIR Waveforms [8]

11.9.1 Interfacing of PIR sensor with Raspberry Pi 2

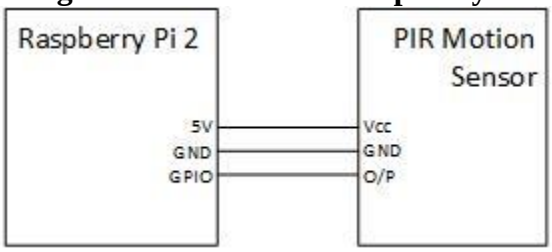


Fig 11.23 PIR Sensor Interfacing

Make the connections as given below

PIR	Raspberry Pi 2
Vcc	+5v
GND	GND
Output	GPIO 4

Output of sensor is **Pi friendly** i.e. 3.3V and it can be powered from the 5V rail of Pi [9].

11.9.2 Programming of Raspberry Pi 2 for PIR sensor

1. Install the GPIO package for Python by following command in terminal:
   $ sudo apt-get install python-dev python-rpi.gpio

2. Write the following Python program in any text editor and save it as PIR.py

```
Import Rpi.GPIO as GPIO          #Import GPIO library
import time                      #Import time library
GPIO.setmode(GPIO.BCM)           #Set GPIO pin numbering
pir = 4                          #Associate pin 26 to pir
GPIO.setup(pir, GPIO.IN)         #Set pin as GPIO in
while True:
   if GPIO.input(pir):           #Check whether pir is HIGH
      print "No Motion"
   else
      print "Motion Detected"
   time.sleep(2)                 #Delay to avoid multiple detection
```

3. Run the program in the terminal by following command
   $ sudo python PIR.py

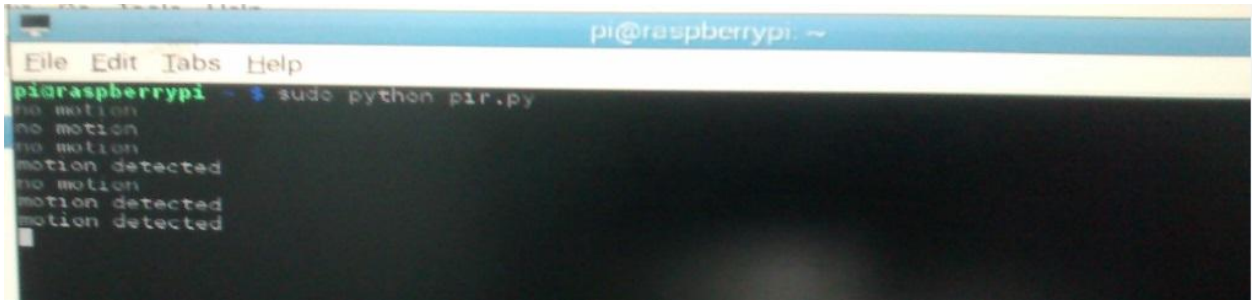


Fig 11.25 PIR Sensor

References

[1] https://learn.adafruit.com/downloads/pdf/dht-humidity-sensing-on-raspberry-pi-with-gdocs-logging.pdf
[2] http://www.micropik.com/PDF/HCSR04.pdf
[3]http://allsensor.in/ProductDetails.aspx?SEOType=BSuwhmB9Ll4%3D&PID=rrb56VpeEIE%3D&CType=ZgDIC937E8s%3D&CID=qcj0eWfX11k%3D
[4] en.wikipedia.org/wiki/Photoresistor
[5] http://www.buildcircuit.com/experiment-with-lm358/
[6] http://www.electro-tech-online.com/threads/op-amp-2-ir-sensors.31350/
[7] en.wikipedia.org/wiki/Passive_infrared_sensor
[8] https://learn.adafruit.com/downloads/pdf/pir-passive-infrared-proximity-motion-sensor.pdf
[9] https://electrosome.com/pir-motion-sensor-hc-sr501-raspberry-pi/

Chapter- 12
Database and Web Services on Raspberry Pi

12.1 Introduction

Database is used to store the data either locally or on the server. For example the received sensor data is stored on local database. There are many databases that support Raspberry pi. Here MySQL is used as a database to store sensor data or any other user data. This section explains how to create, connect database to store real time data for further data analysis and inference.

12.2 Creation of Database

Procedure to create database is as follows:

Step1: Updates the packages using command: **$ sudo get-apt update**
Step2:Install the MySQL database: **$ sudo apt-get install mysql-server python-mysqldb**
Step3: Open the database by using following command and enter password: **$ mysql -u root –p**
Mysql>create database Mydatabase; (Mydatabase is name of the database)
use Mydatabase;
Now for create an account, enter the following commands:
Mysql >create user 'Mymobile'@'localhost' identified by 'password';
Mysql>grant all privileges on abcd.*to 'Mymobile'@'localhost';
Mysql>flush privileges;
Mysql>quit
Step2: Now Mydatabase is created by the user name Mymobile'@'localhost' and password.

Open the mysql database using command:
$ mysql –u Mymobile –p
Password:
Use Mydatabase;
Procedure to create a table in MySQL:
Step1: Run the following command to create the table igdtuw(table name)
Mysql>CREATE TABLE igdtuw(potdata FLOAT,time TIME,date DATE);
If you don't get any error, table by the name IGDTUW is created and can be verified by using mysql command: **Mysql>show tables;**

Procedure to read the sensor datafor potentiometer and saving it into created database.
Step1: Open the leafpad and write the python program as given in 'Program' portion. Save it as name followed by '.py'. For example abc.py
Step2: Open the LX-terminal and run the file abc.py using command: **$ sudo python abc.py**
Running of this file read the data from potentiometer and save it into a local database.
Now you can check your stored data into database.

Program:

```
import spidev
import time
import datetime
import MySQLdb
import RPi.GPIO as GPIO
spi = spidev.SpiDev()
spi.open(0,0)
# read SPI data from MCP3008 chip, 8 possible adc's (0 thru 7)
def readadc(adcnum):
 if not 0 <= adcnum <=7:
return -1
 r = spi.xfer2([1,(8+adcnum)<<4,0])
 adcout = ((r[1]&3) << 8) + r[2]
 return adcout
db = MySQLdb.connect("localhost"," Mymobile","password"," Mydatabase")
curs = db.cursor()
while True:
 val=readadc(0)
print val*0.0032160313
curs.execute("INSERT INTO igdtuw (date,time,potdata) VALUES(now(),now(),'%s')"
,(readadc(0)))
time.sleep(2)
```

12.3 Raspberry Pi as web server

A Web server is a program that uses the client/server model to communicate with World Wide Web's Hypertext Transfer Protocol (HTTP).Apache is a freely available Web server that is distributed under an "open source" license. It is a program that listens for server access requests from Internet browsers/client and grants them if permitted. When you type a URL into your Web browser, Web server replies to the client request.

Procedure to install web server on Raspberry pi.

Step 1:- Update your Raspberry pi by **$sudo apt-get update (update all packages up to date)**
Step 2:- Install apache by command **$sudo apt-get install apache2 php5 libapache2-mod-php5**

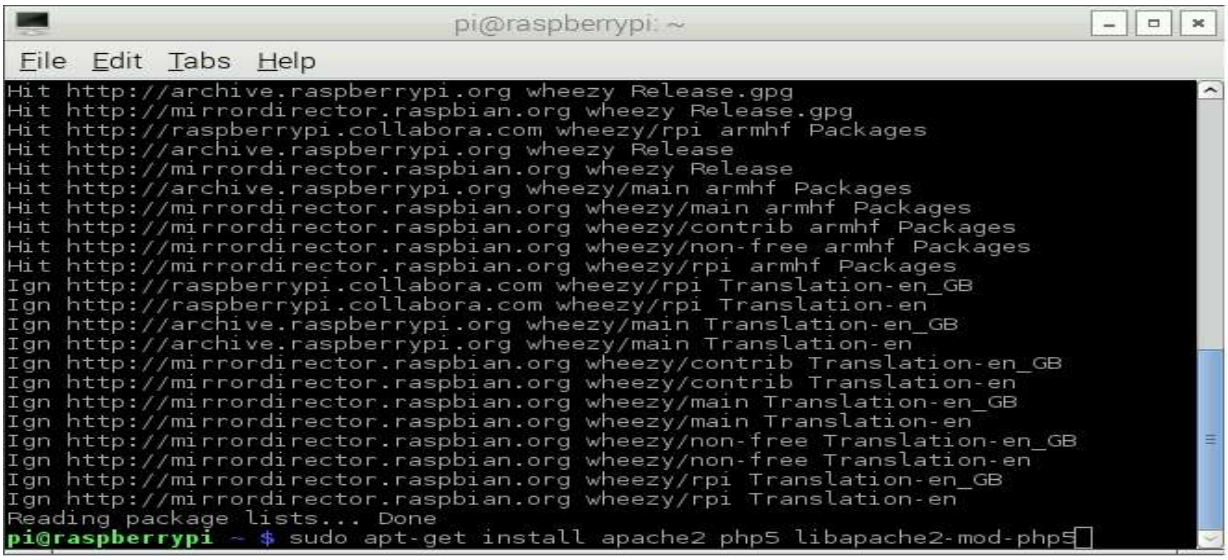


Figure 12.1 install apache server

Step 3:- Restart services of apache server **$sudo service apache2 restart**
Step 4 :- To create a web page go to the WWW directory $cd /var/www/

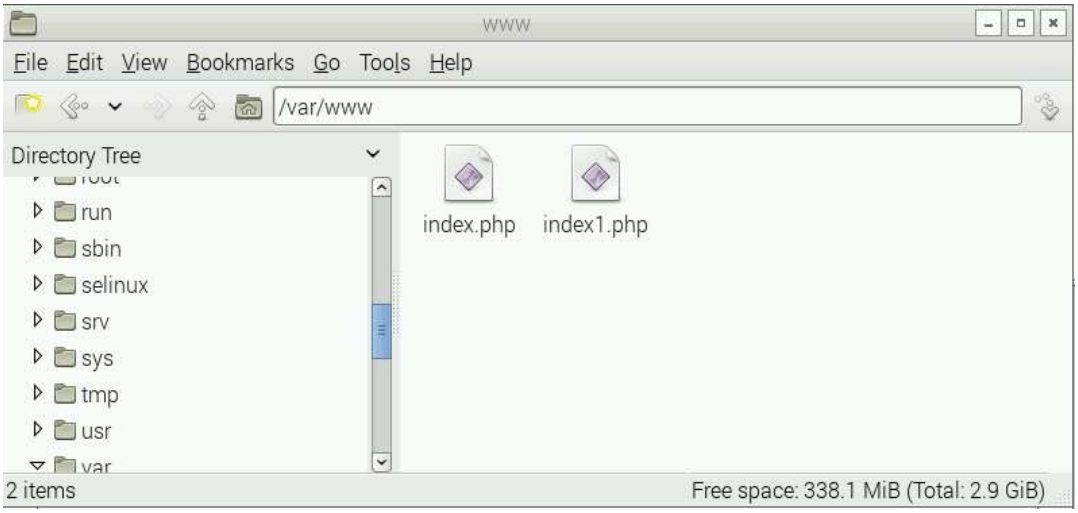


Figure 12.2 Folder Location

Step 5:- Create a PHP file at www folder **$sudo nano index.php**

And enter the following code to create a web page that displays welcome to My smart phone kit, IGDTUW:

```
<?php echo"Welcome  IGDTUW"
echo date ("jS \of F Y h:i:s A") ."<br> ";
?>
```

Step 6:- Type the URL on your web browser with forward slash with file name

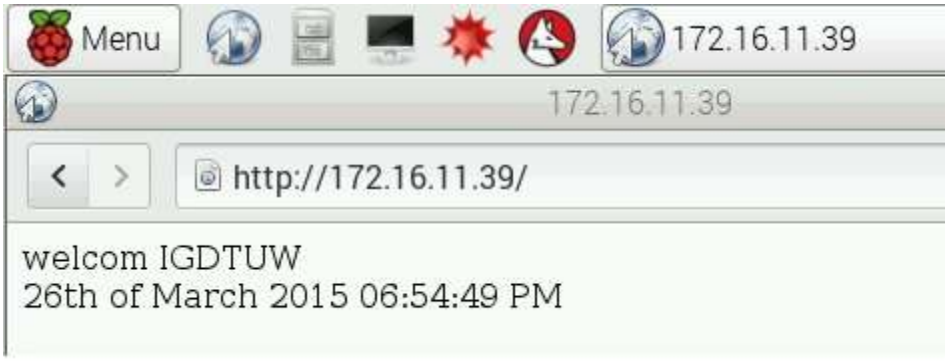


Figure 12.3 Default URL

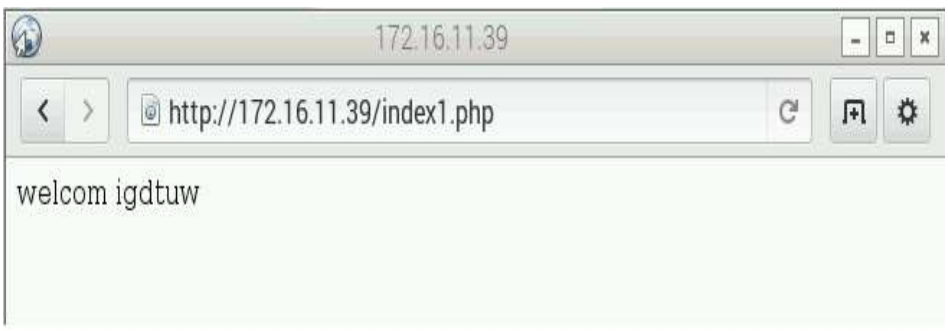


Fig 12.4 URL With File Name

Step 7:- To Resolve your domain name open apache2.conf file in nano by **"$sudo nano /etc/apache2/apache2.conf"** Add the following line at last.
ServerName buildyour.smartfone.com
$sudo service apache2 restart
Display the response on client mobile by typing URL on mobile browser.

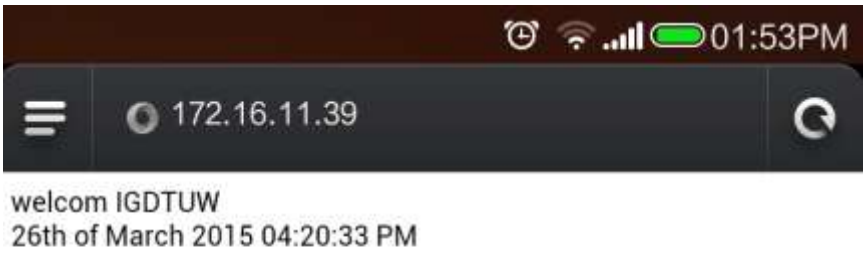


Fig 12.5 web Page over client request

Chapter 13

Raspberry Pi Features

13. 1 Introduction

This chapter explains the various other features of RPi. Rpi can be uses as a linux or unix pc, shell programming, socket programming, c and c ++ programming. It is also being used as a Android / Linux mobile which is explained in previous chapters.

13.2 Unix/Linux Command Reference: Various commands that can be used to experiment are listed here. File Commands

1.ls	Directory listing
2.ls -al	Formatted listing with hidden files
3.ls -lt	Sorting the Formatted listing by time modification
4.cd dir	Change directory to dir
5.cd	Change to home directory
6.pwd	Show current working directory
7.mkdir dir	Creating a directory dir
8.cat >file	Places the standard input into the file
9.more file	Output the contents of the file
10.head file	Output the first 10 lines of the file
11.tail file	Output the last 10 lines of the file
12.tail -f file	Output the contents of file as it grows, starting with the last 10 lines
13.touch file	Create or update file
14.rm file	Deleting the file
15.rm -r dir	Deleting the directory
16.rm -f file	Force to remove the file
17.rm -rf dir	Force to remove the directory dir
18.cp file1 file2	Copy the contents of file1 to file2
19.cp -r dir1 dir2	Copy dir1 to dir2; create dir2 if not present
20.mv file1 file2	Rename or move file1 to file2,if file2 is an existing directory
21.ln -s file link	Create symbolic link link to file

For Process management

22.ps	To display the currently working processes
23.top	Display all running process Unix/Linux Command Reference
24.kill pid	Kill the process with given pid
25.killall proc	Kill all the process named proc
26.pkill	pattern Will kill all processes matching the pattern
27.bg	List stopped or background jobs, resume a stopped job in the background
28.fg	Brings the most recent job to foreground
29.fg n	Brings job n to the foreground

For File permissions

1.chmod octalno filename

Searching

1.grep	pattern file Search for pattern in file
2.grep -r	pattern dir Search recursively for pattern in dir
3.command | grep	pattern Search pattern in the output of a command
4.locate file	Find all instances of file
5.find	Searches in the current directory (represented by a period)
6.pgrep pattern	Searches for all the named processes

System Info

1. date	Show the current date and time
2.cal	Show this month's calendar
3.uptime	Show current uptime
4.w	Display who is on line
5.whoami	Who you are logged in as Unix/Linux Command Reference
6.finger user	Display information about user
7.uname -a	Show kernel information
8.cat /proc/cpuinfo	Cpu information
9.cat proc/meminfo	Memory information
10.man command	Show the manual for command
11.df	Show the disk usage
12.du	Show directory space usage
13.free	Show memory and swap usage
14.whereis app	Show possible locations of app
15.which app	Show which applications will be run by default

Compression

1.tar cf file.tar file	Create tar named file.tar containing file
2.tar xf file.tar	Extract the files from file.tar
3.tar czf file.tar.gz files	Create a tar with Gzip compression
4.tar xzf file.tar.gz	Extract a tar using Gzip
5.tar cjf file.tar.bz2	Create tar with Bzip2 compression
6.tar xjf file.tar.bz2	Extract a tar using Bzip2
7.gzip file	Compresses file and renames it to file.gz
8.gzip -d file.gz	Decompresses file.gz back to file

Network

1.ping host	Ping host and output results
2.whois domain	Get who is information for domains
3.dig domain	Get DNS information for domain
4.dig -x host	Reverse lookup host
5.wget file	Download file
6.wget -c file	Continue a stopped download Unix/Linux Command Reference

Shortcuts

1.ctrl+c Halts the current command
2.ctrl+z Stops the current command, resume with fg in the foreground or bg in the background
3.ctrl+d Logout the current session, similar to exit
4.ctrl+w Erases one word in the current line
5.ctrl+u Erases the whole line
6.ctrl+r Type to bring up a recent command
7.!! Repeats the last command
8.exit Logout the current session

13.3Shell script

Unix-style architecture for operating systems is used to experiment the shell programming. The command-line interface is known as the shell. Te shell environment helps users to interact with and access core functions of the operating system. Execution of a script can be done in two ways. Either we can run the script as a command-line argument to bash or we can grant execution permission to the script so it becomes executable. Write the following code in a file named **arith.sh** in home folder, change the permission of file using command **$ chmod -x arith.sh** and execute it at command line as **./arith.sh**

1 Simple program Arithmetic expression

```
#!/bin/sh
# Perform some arithmetic
x=24
y=4
Result=”expr $x \* $y”
echo "$x times $y is $Result"
```

2. Translating Characters

converting one character to another,
translating and saving string stored in a variable. Store the following in a file named **tr1.sh** and execute it

```
#!/bin/sh
# Translate the contents of a variable
Cat_name="Piewacket"
echo $Cat_name | tr 'a' 'i'
```

3. find hostname of your PC

```
#!/bin/ksh
echo Today is `date`
file=/etc/hosts
echo The file $file has $(wc -l < $file) lines
hostname -s > myhostname
echo This system has host name $(<myhostname)
```

4. do while loop count the number

```
#!/bin/ksh
 count=0
 max=10
 while [[ $count -lt $max ]]
 do
 echo $count
 count=$((count + 1))
 done
```

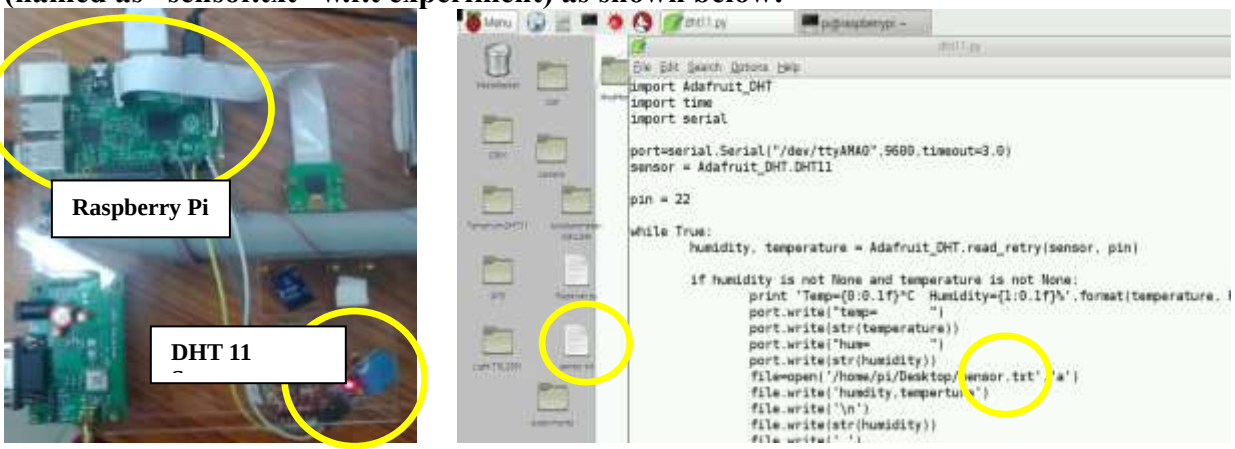
5. Reading a file line by line.

```
#!/bin/sh
# Demonstrate reading a file line-by-line, using I/O
 # redirection in a compound command
 # Also test variable setting inside an implicit subshell.
 # Test this under sh and ksh and compare the output.
line="TEST"
save=
if [ -z "$1" ]; then
        echo "Usage: $0 filename"
else
        if [ -r $1 ]; then
                while read line; do
                        echo "$line"
                        save=$line
                done < $1
        fi
fi
echo "End value of \$line is $line"
echo "End value of \$save is $save"
```

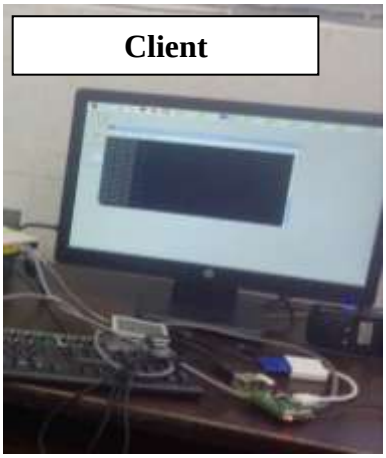
13.4 Socket Programming

The procedure for establishing the connection between the server and client, and communication between them through socket are explained here.

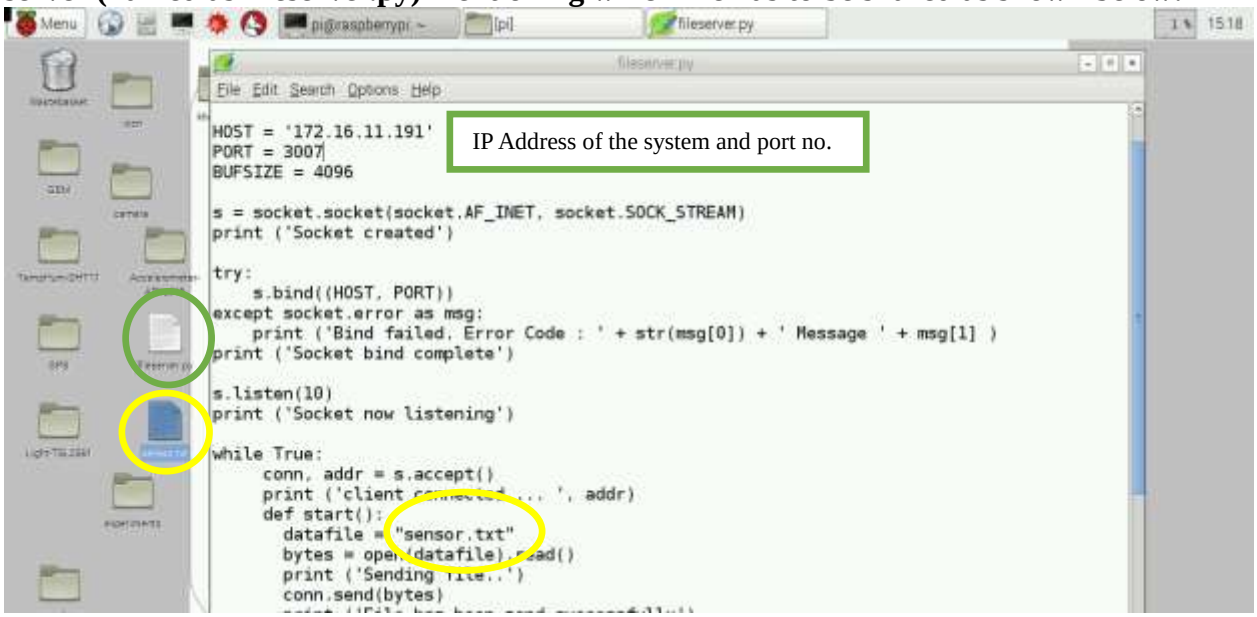
STEP 1: Connect the Raspberry Pi to DHT11 sensor and store the real time readings to a file (named as "sensor.txt" w.r.t experiment) as shown below:



Scenario of sharing a real time sensor readings file remotely and receiving the file on another raspberry pi through socket programming:



STEP2: Connect the Raspberry Pi to internet and then open leafpad and write a program for server (named as fileserver.py) mentioning which file has to be shared as shown below:

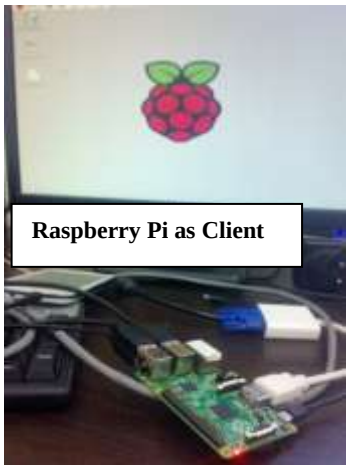


CODE FOR THE SERVER SOCKET:

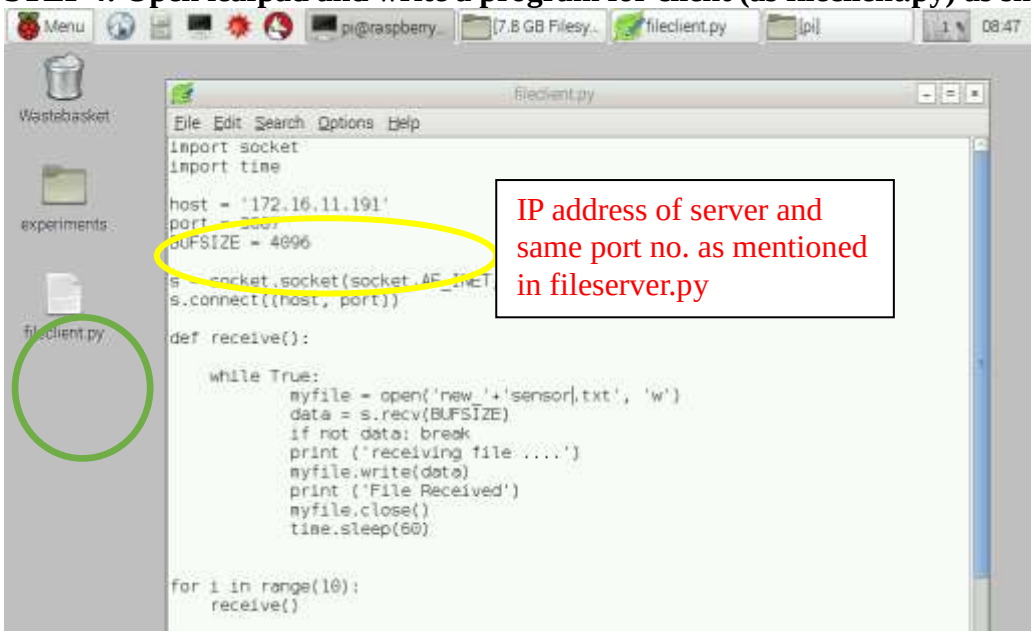
CODE:

```
import socket
import time

HOST = 'ip address of the system'
PORT = port address
BUFSIZE = 4096
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print ('Socket created')
try:
   s.bind((HOST, PORT))
except socket.error as msg:
   print ('Bind failed. Error Code : ' + str(msg[0]) + ' Message ' + msg[1] )
print ('Socket bind complete')
s.listen(10)
print ('Socket now listening')

while True:
    conn, addr = s.accept()
    print ('client connected ... ', addr)
    def start():
      datafile = "server.txt" #filename which you want to send#
      bytes = open(datafile).read()
      print ('Sending file..')
      conn.send(bytes)
      print ('File has been send successfully')
      time.sleep(50)

    for i in range(10):
```

STEP 3: Setup another Raspberry Pi on which real time sensor readings file has to receive remotely and connect it with the internet.



STEP 4: Open leafpad and write a program for client (as fileclient.py) as shown below:



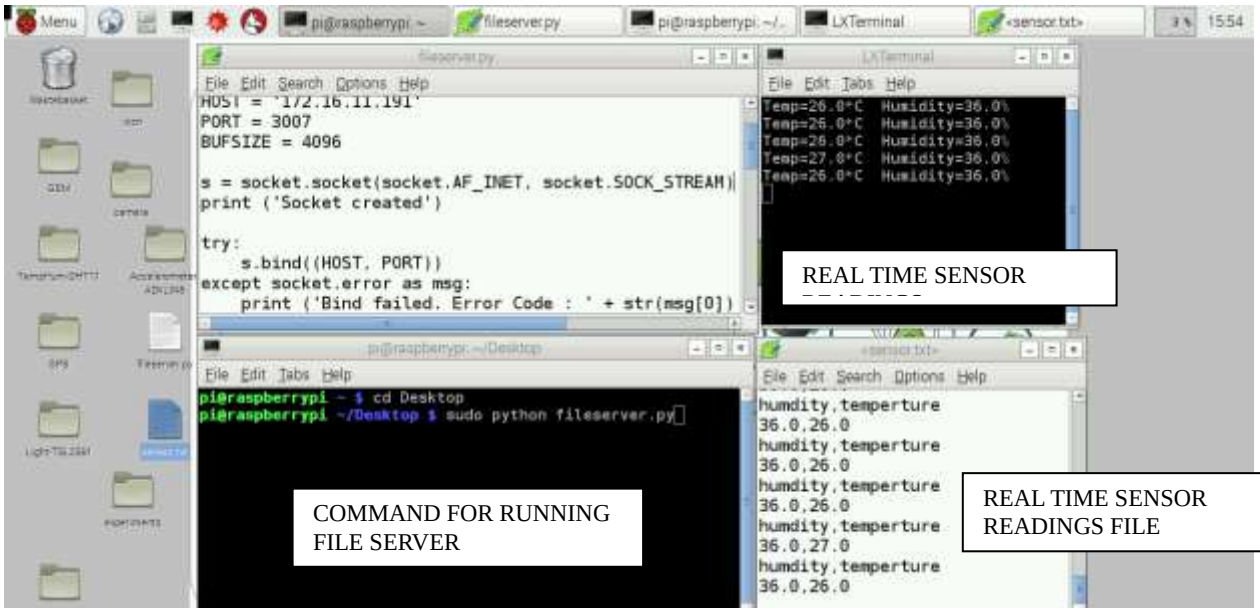
CODE FOR CLIENT:

CODE:

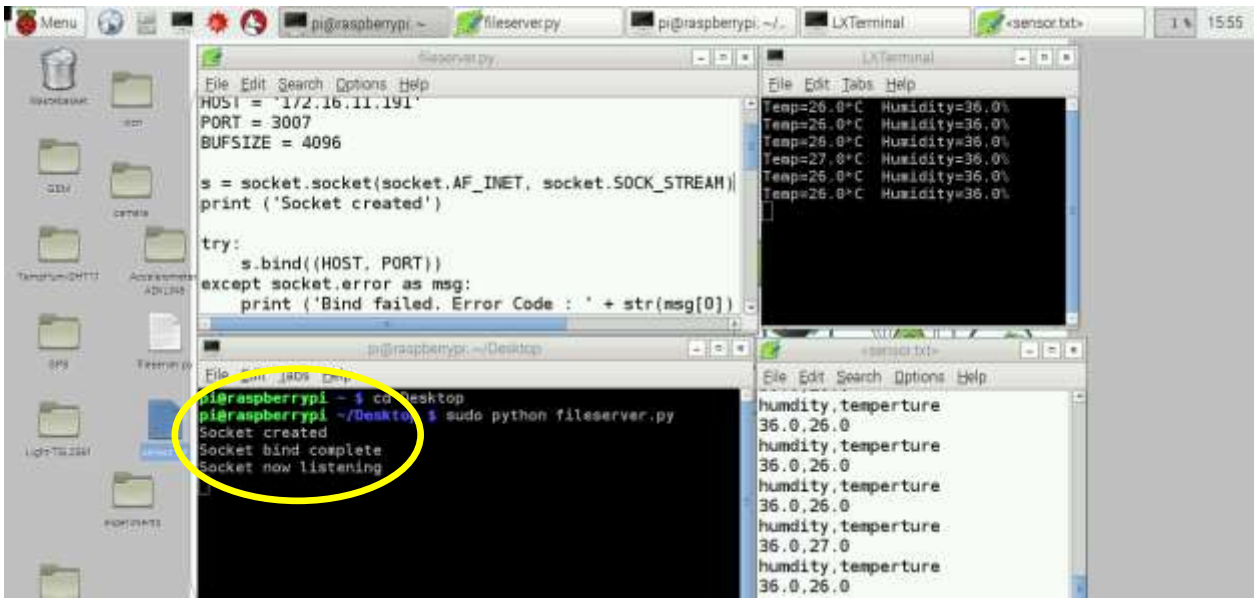
```
import socket
import time
host = 'ip address of the server'
port = port address as mentioned in the server
BUFSIZE = 4096
s = socket.socket(socket.AF_INET, socket.
s.connect((host, port))

def receive():
    while True:
        myfile = open('new_'+'sensor.txt', 'w')
        data = s.recv(BUFSIZE)
        if not data: break
        print ('receiving file ....')
        myfile.write(data)
        print ('File Received')
        myfile.close()
        time.sleep(60)

for i in range(10):
```

The file received from the server will be saved as new_sensor.txt

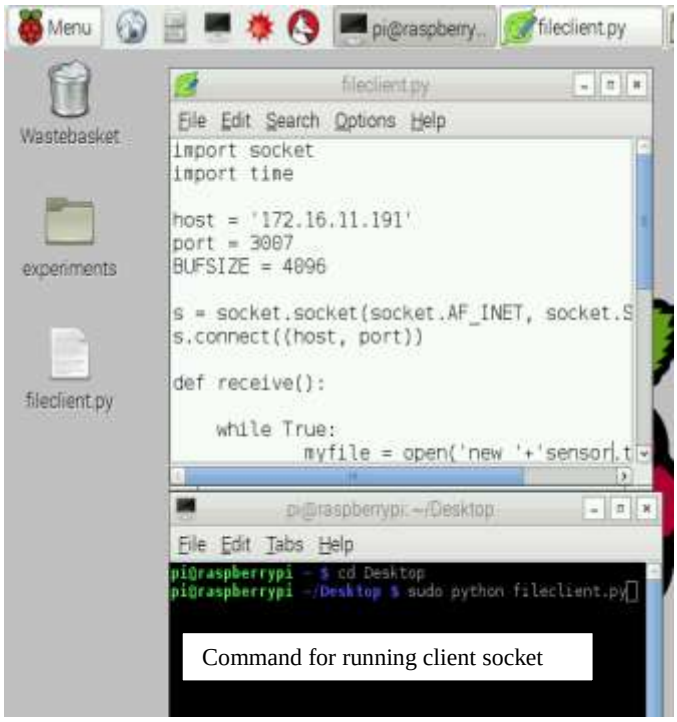
the client:



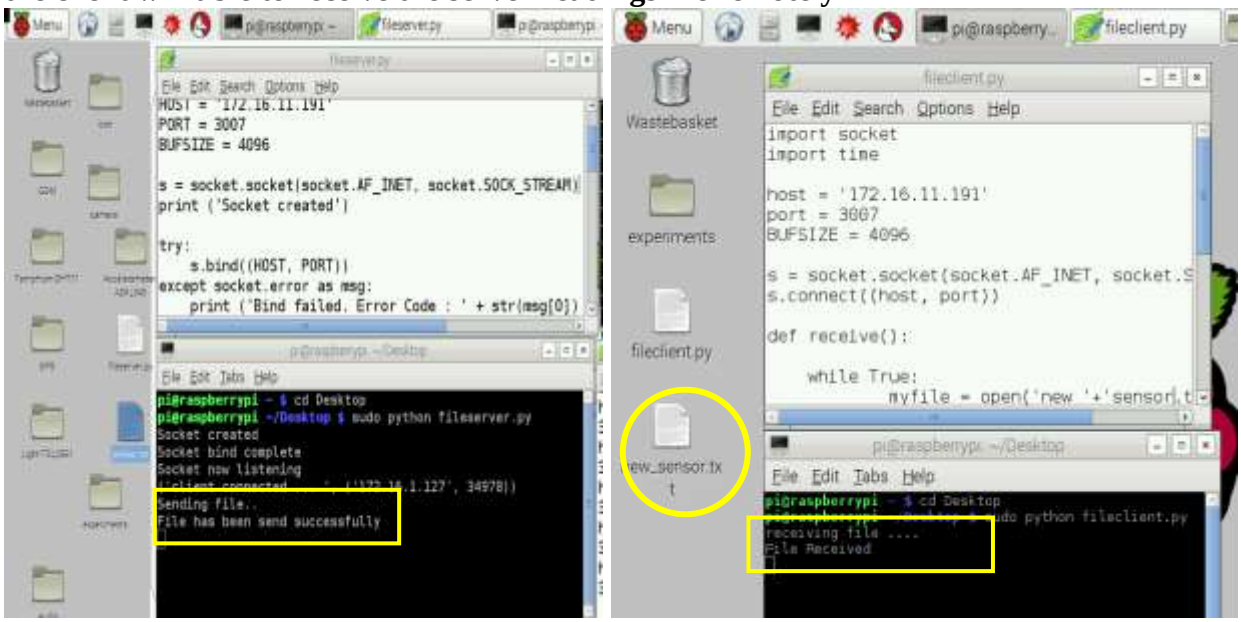
Note: when fileserver.py will run successfully then socket will be created and it is ready to listen to the other socket connection as shown below:



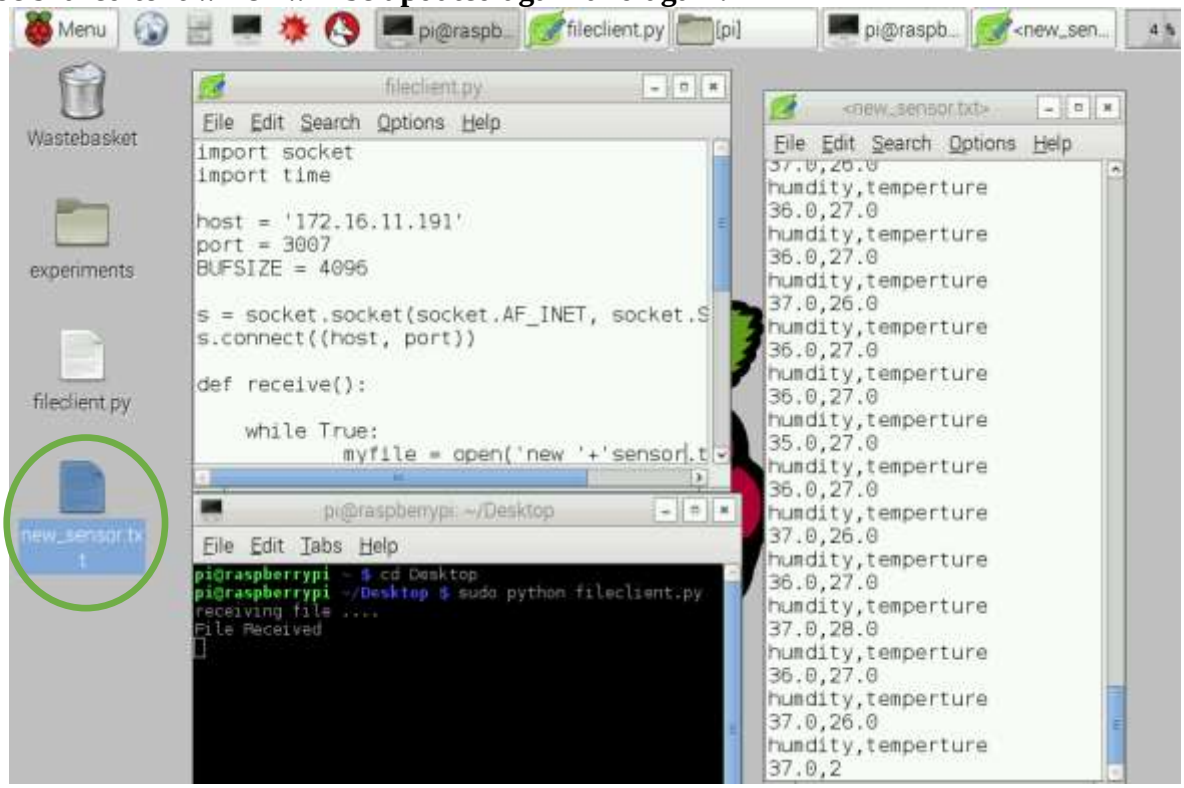
STEP 6: Go to client and open LX terminal and run the client code which will create socket and listen to server



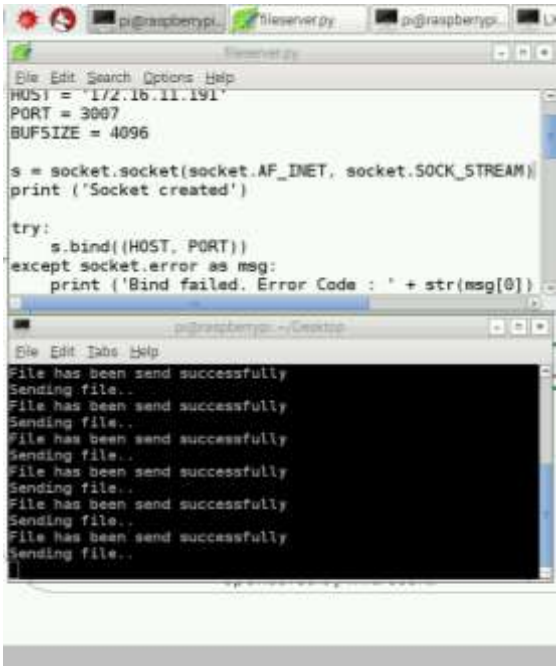
STEP7: When the client will run then the client socket will be connected to server socket and the client will able to receive the server readings file remotely

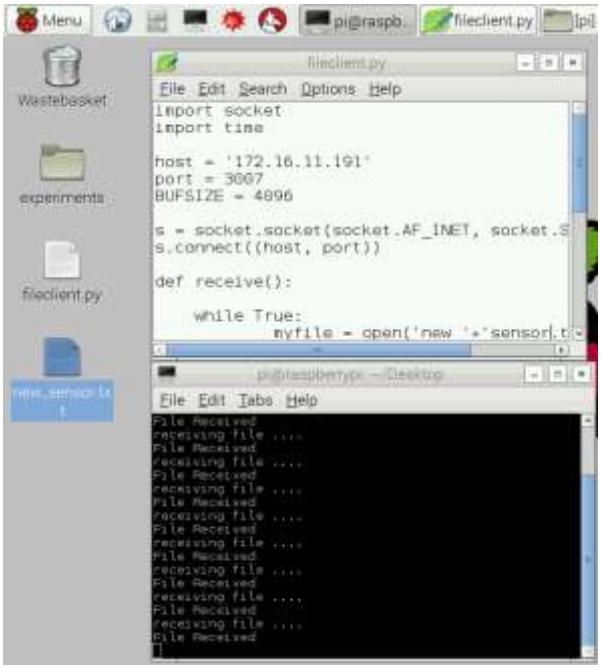


STEP 8: When client code will run it will establish the connection with server and the file will be shared to it which will be updated again and again.



REAL TIME UPDATION OF THE SHARED FILE





INSTRUCTIONS:

1. **THE IP ADDRESS AND PORT NUMBER MENTIONED IN THE CLIENT SHOULD BE SAME AS SERVER**

Appendix A:Parts and suppliers		
Essential	Component & Specification	Web Links –Approved by funding agency
	Raspberry Pi 2	http://in.rsdelivers.com/product/raspberry-pi/raspberry-pi-b/raspberry-pi-b/8111284.aspx
	Touchscreen (TFT 3.5",320x480)	http://www.tinylcd.com/tiny149/product.php?id_product=37
	Camera (5MP)	https://www.crazypi.com/Wireless-Usb-security-camera-raspberry-pi?search=raspberry%20pi%20camera
	GSM Module (SIM900)	http://www.ventor.co.in/index.php?main_page=product_info&products_id=181
	Bluetooth Module (HC-05)	http://www.ventor.co.in/index.php?main_page=product_info&cPath=68_48&products_id=254
Additional	GPS Module (TTL UART - Patch Antenna on Top)	http://sunrom.com/p/gps-receiver-ttl-uart-patch-antenna-on-top
	Temp. and Humidity sensor Module (DHT-11)	http://www.ventor.co.in/index.php?main_page=product_info&products_id=289
	Accelerometer Module (ADXL345)	http://www.ventor.co.in/index.php?main_page=product_info&products_id=303
	Light Sensor (TSL2561 Module)	http://www.ebay.in/itm/Sparkfun-TSL2561-Luminosity-Sensor-Breakout-Module-/201110320032
Accessories	Power Adaptor (5V/2A, 230V AC, 50Hz)	http://www.flipkart.com/htc-5v-2a-10w-battery-charger/p/itmdzcz9gsd8tzkt?pid=ACCDZCZ9TXZMJYZZ&srno=b_7&ref=f293c247-7486-4947-a900-112c6d475451
	SD card (16gb) class 10	http://www.snapdeal.com/product/sandisk-mobile-ultra-8-gb/257968#bcrumbLabelId:228
	Keyboard & Mouse(during development), HDMI Cable, WiFi etc	http://www.snapdeal.com/product/iBallWinne/34019#bcrumbSearch:|bcrumbLabelId:152
	ADC add on board with LCD etc	www.trinitymicrosystem.com

Smart Phone Kit is Manufactured &Marketed By:

Dr. M Y Rathnam, MD

Trinity Microsystems Pvt. Ltd

313 Triveni Complex ,E-10-12,Jawahar Park, Vikas Marg, Laxmi Nagar.**NEW DELHI-110092,(INDIA)**, Fax:-91-11-22541518, Phone:-011-22016186, 011-22458310, Mobile no :-+91-9810190925

Website:www.trinitymicrosystem.com, E-mail:sales@trinitymicrosystem.com

E-mail :dli@trinitymicrosystem.com

Please refer for more details at: http://mysmartphonekit.mobileeducationkit.net/

www.mobileeducationkit.net

ABOUT IGDTUW

Indira Gandhi Delhi Technical University for Women (IGDTUW) has been upgraded from Indira Gandhi Institute of Technology in May 2013 vide Delhi State Legislature Act 9, 2012, as a non-affiliating Teaching and Research University at Delhi to facilitate and promote studies, Research, Technology, Innovation, Incubation and extension work in emerging areas of professional education among women, with focus on Engineering, Technology, applied sciences, management and its allied areas with the objective to achieve excellence in these and related fields.

www.igdtuw.ac.in

www. mysmartphonekit.mobileeducationkit.net

ABOUT TRINITY

Trinity Microsystems Pvt Ltd started in 1996 in New Delhi with the Manufacturing of Electronics/Electrical Lab Equipments covering Digital Electronics, Analog Electronics, Communication, Microwave, Fiber Optics, Embedded Technology, Power Electronics, Drives, Instrumentation, Control, Power System Simulators, Robotics, Bio & Nano Technology and Solar/Wind Energy etc Training Equipments and Simulation Softwares, supplying to many Govt/Pvt. Engg Colleges, universities, Polytechnics etc in India and abroad.

INDIAN PATENT FILED

A System for Building, Customizing Software and Hardware Interfaces of Smart Phone using Raspberry Pi, Application No. 2354/DEL/2015

IGDTUW Transferred/ Licensed Technology To Trinity Microsystems Pvt. Ltd, New Delhi (india)

ACCESSORIES

CD, User's Manual ,Online support @ www.mysmartphonekit.mobileeducationkit.net

My_SmartPhone kit based on Intel Edison, Intel Galileo, Gen2, Intel Atom , Snapdragon are to be released soon

MANUFACTURED & MARKETED BY

TRINITY MICROSYSTEMS PVT. LTD

313, Triveni Complex, E-10-12, Jawahar Park, Vikas Marg, Laxmi Nagar

NEW DELHI-110092 (INDIA)

Phone : 011-22016186, 011-22458310,

Fax : 91-11-22016186 (D) 22467614

Contact Person : Dr. M Y Rathnam, MD

Mobile No. : +91-9810190925

Website:www.trinitymicrosystem.com

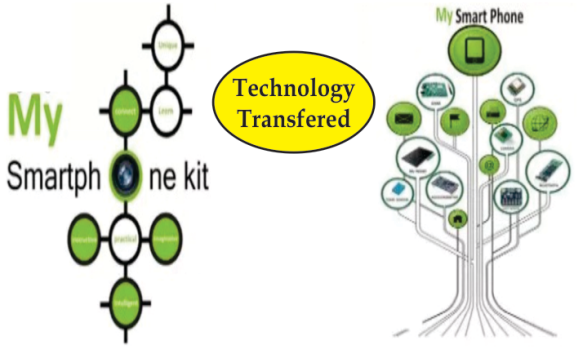
E-mail : sales@trinitymicrosystem.com

: dli@trinitymicrosystem.com

Trinity

AN ISO 9001-2008 Certified

My_Smart Phone Kit



SPONSORED BY

Microsoft Mobile University Relations (Nokia University Relations) and Association with MHRD(DIC&ITRA)

DEVELOPED BY

Deptt. of Computer Science and Engineering, Indira Gandhi Delhi Technical University for Women Under the aegis of MY_SMARTPHONE Project

Dr. SRN Reddy (PI), HoD CSE, IGDTUW,

Kashmere Gate, Delhi-110006

Tel.: 011-23900100/253, 9810101742

Email: srnreddy@igdtuw.ac.in

Website: www.mobileeducationkit.net

MY_SMART PHONE KIT

- The kit is Designed to be used for M.Tech, B.Tech, MCA, BCA, MSc, Diploma courses of CSE/IT/ECE/EEE etc as a Practical experimental Setup for various courses.
- The Kit can be used as a Linux and Android PC to Experiment the theoretical concepts in practical/real environment.
- It provides the freedom in Hardware interfaces, building the coustomizable OS and integrating the applications with the H/W and OS as a complete system solution.
- Several commercial & Innovative projects are also provided to facilitate the capabilities and learning methodologies for faculty, students and developers further to developed innovative off the shelf projects.
- It uses completely open source hardware, software & developmental tools to provide the integrated solution along with online portal, textbook that describes various experiments of different subjects in detail.
- This kit explores the integration of application development with open source customized OS (Linux & Android) to prototype the project ideas and experiments. Components will be made available as per the needs of projects and requirements of any application.

Interested faculties and students can contribute for further research and the same will be published after review.

MY_SMART PHONE KIT

Fig. *MY_SMART PHONE KIT*

FEATURES

Several aspects of OS, Communication, Sensors are provided to explain various aspects practically and theoretically.

MOBILE OS

1. Linux based (customized)
2. Android based (customized)
3. Windows based (customized)

COMMUNICATION

1. Wired (SPI,I^2C, UART,USB)
2. Wireless (GSM,GPS, Bluetooth, Wi-Fi)

I/O INTERFACES

Input & Output peripherals such as Mike, Speaker, Camera, TFT etc are

SENSOR'S INTERFACE

Several sensors like **Temperature, Humidity, Light, Accelerometer,** etc. are interfaces to model the real-world systems along with the database to provide the useful information via communication interfaces and mobile apps.

COURSES SUPPORTED

The Kit is designed to support the following courses fully or partially for students/ faculty:-

1. Mobile Computing
2. Embedded System
3. Wireless Communications
4. Design & Development of Mobile Devices
5. Smart Device Development
6. Socket Programming
7. Internet Of Things
8. Programming Languages such as Python, C, Shell Script etc
9. Real-time OS and Databases etc

Suitable for practical applications and system development for M.Tech, B.Tech, MCA, BCA, MSc & Diploma courses etc.

BENEFICIARIES

Faculty who are teaching Communication Systems, Mobile Computing, Embedded Systems , students and hobbyists .

COLLABORATING INSTITUTES

IIT Delhi, IIIT Delhi, NITTTR, Chandigarh, NIT Warangal etc.

TRAINING

Resource Persons from IGDTUW, TRINITY MICROSYSTEMS PVT. LTD, NEW DELHI can provide the training based on the requirements.

FEEDBACK

Share your feedback / suggestions on
www. mysmartphonekit.mobileeducationkit.net
dli@trinitymicrosystem.com

About the Authors

Dr. S. Ramanarayana Reddy is working as a Head of the CSE Dept in Indira Gandhi Delhi Technical University for Women (IGDTUW), Delhi. He has awarded PhD in the area of Embedded Systems Design in 2009 and M. Tech degree in 2002 from Jawaharlal Nehru University, New Delhi. His research interest includes Embedded Systems Design, Mobile Architecture and Programming, Systems Programming, and Real-Time Systems. He has published more than 30 research papers in various international journals and conferences and filed couple of patents.

He is a PI in five externally funded sponsored projects by Intel, Microsoft and Nokia and Co PI for two MHRD sponsored projects in collaboration with IIT Delhi and IIIT Delhi. He established the state of art Embedded & Mobile Design and Innovation lab in collaboration with Nokia, Intel, Microsoft, Atmel, ARM etc. to provide the quality education and research with a practical environment. He has conducted several training programs on 'Mobile Architecture and Programming' and Embedded Systems for Industry, faculty members and students. He has more than 15 years experience that includes research, industry and teaching. He is a member of CSI, VLSI Society of India and AMIE and guiding M.Tech and PhD students.

Suresh Chande has spent close to 2 decades in the Mobile industry in the Mobile Applications and Services domain covering Telecommunications and Computer science field. He has been over 10 years Nokia Research Center leading research and development on Mobile applications and services covering areas of mobile applications and user research in the space of productivity and new innovative and collaborative applications. His work has led to several patents and applications deployed into millions of smartphones. He has extensive knowledge in product making and has worked in Mobile Product development. He has worked in Linux or MeeGo based N9, Android based Nokia X, Windows Phone and several Symbian operating system based Mobile phone products. He has worked with multiple open source projects which has deployed onto Mobile device such as N9 which has been recognized and gained best in class Smartphone platform. He works for Microsoft Mobile devices group in Microsoft after part of Nokia Devices and Services business was acquired by Microsoft.

Suresh Chande is a Global Head of Consumer Data in Consumer Engagement, Digital Marketing Organization in Microsoft Mobile Devices group in Microsoft. At Digital marketing organization he is responsible for Consumer Data Strategy to leverage Data Science and Big Data technologies across multiple digital channels covering Mobile Devices, Social Networks and Online .COM web site.

Suresh Chande with his deep research background also supports Microsoft Mobile as the Head of a Global Academic Relations Program. He is responsible to engage academic partners, students and researchers to research and collaborate with Microsoft Mobile. He works closely with a large network of universities across India and Finland. As a researcher and his passion to innovate he continues his research work on utilization of Workflows technologies and automating mobile interactions with existing business process automation systems.

For Books on following Subjects

- Business, Management & Finance
- Career Development, Career Guides
- Catering & Hotel Management / Recipes
- Civil Engineering
- Computers
- Communication
- Dental / Health / Medical
- Economics
- Electrical Engineering
- Electronics & Communication
- English
- Entrepreneurship
- Environmental Engineering
- Event Management
- Fiction
- Forensic Science
- General Titles
- HRD
- International Trade
- Law
- Learning Disability
- Mathematics
- Mechanical Engineering
- Media
- Mobile Computing
- Motivation & Self Help
- Parenting
- Patent
- Physics
- Project Management / Software Engineering
- Real Estate
- Statistics

please visit

shroffpublishers.com

Publishers We Represent

THE X TEAM

IBM Redbooks

O'REILLY®

Rheinwerk Publishing

SAP PRESS

AGILITEK

Pragmatic Bookshelf

MAKERMEDIA

T. F. Wallace

MURACH

RAMPANT

sitepoint

SVP PRESS

RMC

Munir Hamad

MC PRESS

Pulp Free Press

Janson Industries

IT Governance Publishing

CRC PRESS

bizle PROFESSIONAL SERIES

DVT Press

nova vista

J.ROSS PUBLISHING

For Wholesale enquiries contact:-

SHROFF PUBLISHERS & DISTRIBUTORS PVT. LTD. SPD®

C-103, TTC Industrial Area, MIDC, Pawane, Navi Mumbai - 400 703 • TEL: (91 22) 4158 4158 • FAX: (91 22) 4158 4141
E-mail : spdorders@shroffpublishers.com • Web : www.shroffpublishers.com • CIN : U22200MH1992PTC067760

Branches:-

Bangalore
7, Sharada Colony, Basaveshwarnagar,
8th Main, Bangalore 560 079
Tel: (91 80) 4128 7393 • Fax: 4128 7392
E-mail: spdblr@shroffpublishers.com

Delhi
Basement, 2/19 Ansari Road,
Daryaganj, New Delhi - 110 002
Tel: (91 11) 2324 3337 / 8 • Fax: 2324 3339
E-mail: spddel@shroffpublishers.com

Kolkata
7B Haati Bagan Road,
CIT Paddapukur, Kolkata - 700 014
Tel: (91 33) 2284 9329 / 7954 • Fax: 2835 0795
E-mail: spdkol@shroffpublishers.com

Mumbai
36, M. A. Sarang Marg,
(Tandel Street South) Dongri, Mumbai-400 009.
Tel.: (91-22) 6610 7595 • Telefax: 6610 7596
E-mail:spddongri@shroffpublishers.com

RESIDENT REPRESENTATIVES

Chennai Mobile : 09710936664 / 09884193326 E-mail: spdchennai@shroffpublishers.com

Nagpur Mobile: 07709504201 Email: rajendra@shroffpublishers.com

Pune Mobile : 09850446647 E-mail: umesh.spd@gmail.com

For retail enquiries contact:-

COMPUTER BOOKSHOP www.cb-india.com

Computer Bookshop (I) Pvt. Ltd.
Kitab Mahal Bldg., Ground Floor, Next to Central Bank of India.
190, Dr. D. N. Road, Fort, Mumbai 400 001
Tel: (91 22) 6631 7922 / 23 / 24 • Fax: 2262 3551
E-mail: info@cb-india.com

ESTAB. 1935®

Sterling Book House
181, Dr. D. N. Road, Fort, Mumbai - 400 001.
Tel. : (91-22) 2267 6046, 2265 9599 • Fax : 2262 3551
E-mail : sbh@vsnl.com • www.sterlingbookhouse.com

BOOKZone
BOOKSHOP FOR PROFESSIONALS

Shop #1, Cassinath Building, 172/174,
Dr. D. N. Road, Mumbai - 400 001.
Tel. : (91-22) 2205 4616/17 • Fax: 2205 4620
E-mail : mail@bookzone.in • www.bookzone.in

About the book

- A integrated approach includes software customization, Hardware Interfacing along with the book
- Set of experiments and projects are made available through online for further reference.
- Online web support for both hardware, software and application development
- Practical approach with real hardware and software
- Use of open source language, operating system and hardware which is accepted across the globe for wide applications and courses
- This book serves as a practical environment to perform the experiments related to mobile computing, mobile communications, computer architecture and embedded systems etc.

ISBN: 978-93-5213-281-2

SHROFF PUBLISHERS & DISTRIBUTORS PVT. LTD.