IoT BASED DATALOGGER SYSTEM

(WEATHER MONITORING USING RASPBERRY PI)

A mini project report submitted to ECIL in partial fulfilment of the requirements for the award of the degree

Bachelor of Technology

in

Electronics and Communication Engineering

submitted by

BEDIKHANNA PAVAN KULKARNI-1724A04P0

SRIRAM SAI KRISHNA -17241A04T4

T. RAUNAK KUMAR -17241A04T5

YADAVALLI VIVEK-17241A04U0

Gokaraju Rangaraju Institute of Engineering & Technology

Nizampet Road, Bachupally, Kukatpally

Hyderabad- 500090, Telangana State, India.

Electronics Corporation of India Limited

ELECTRONICS CORPORATION OF INDIA LIMITED

CERTIFICATE

This is to certify that B. Pavan Kulkarni, S. Sai Krishna, T. Raunak Kumar and Y. Vivek bearing Reg. no's 17241A04P0, 17241A04T4, 17241A04T5 and 17241A04U0 student of third year in **Electronics and Communication Engineering, Gokaraju Rangaraju Institute of Engineering & Technology, Bachupally,** have completed the mini project work titled **"IoT BASED DATA LOGGER SYSTEM"** at CED, ECIL from 8th May 2019 to 7th Jun 2019.

They are sincere hardworking and had studied with excellent enthusiasm in particular aspects. Their conduct during the course of training at organization is found good.

IN-CHARGE CED

CONTENTS

5. BMP180 DIGITAL PRESSURE SENSOR

6. 16 x 2 CHARACTER LCD

7. RASPBERRY PI WITH RASPBIAN OS

8. ThingsSpeak and IoT

9. Results

ABSTRACT

Climatic change and environmental monitoring have received much attention recently. Man wants to stay updated about the latest weather conditions of any place like a college campus or any other particular building. Since the world is changing so fast so should the weather stations. This weather station is based on IoT (Internet of Things). These systems are designed to monitor vital physical phenomena generating data which can be transmitted and saved at cloud from where this information can be accessed through applications and further actions can be taken.

It is equipped with environmental sensors used to capture distributed meteorological measurements like temperature, humidity, altitude and pressure at any particular place and report them in real time on cloud. To accomplish this, we used Raspberry PI3 and different environmental sensors like DHT11, BMP180. The sensors constantly sense the weather parameters and keeps on transmitting it to the online web server over a Wi-Fi connection. The system employs different sensors such as DHT11 and BMP180 which transmits data to open IoT API service ThingSpeak where it is analysed and stored. The weather parameters are uploaded on the cloud and then provides the live reporting of weather informatics. This project also focuses on the IoT application in the new generation of environmental informatics and provides a new paradigm for environmental monitoring in future. It will also give the graphical representation of the weather parameters that will help the user to compare the weather statics of different instants of time and from this graphical representation the user can predict the weather of that particular place.

CHAPTER-1

INTRODUCTION TO EMBEDDED SYSTEMS

History:

Now a days we observe that, the term weather change is growing in preferred use to 'Global warming' because it helps to convey that there are other changes in addition to rising temperature. Climate change refers to any significant change in measures of climate (such as temperature, precipitation or wind) lasting for an extended period (decades or longer).

Like most other developing countries of Africa, Kenya's rain-fed agriculture is the backbone of the economy. As such, the Country's economy and weather are so intertwined that a good season means healthy economy and famine, and death otherwise. So, it becomes important to monitor the weather.

Here we have come up with a system which monitors the weather which is also called a weather station. A weather station is a facility, either on land or sea, with instruments and equipment for measuring atmospheric conditions to provide information for forecasts and to study the weather and climate. The measurements taken include temperature, atmosphericpressure, humidity, windspeed, winddirection, and precipitation amounts. Wind measurements are taken with as few other obstructions as possible, while temperature and humidity measurements are kept free from direct solar radiation, or insolation.

Overview:

The aim of this project is to create an online weather system which enables a user to check real-time weather parameters of a place anytime and anywhere with just a few buttons click. In this project, a weather station will be built to collect weather parameters. Data collected will then store into Cloud Storage.

Here we are thingsspeak web server. After storing the data, analysis of weather will be done. A web channel will be developed to display the real-time weather conditions in form of graphs. Also, the system is integrated with a 16x2 LCD Display on which the temperature humidity and pressure will be displayed.

By this system there two main advantages:

- First is that the data will be displayed in the cloud storage and also in LCD so one can easily know the real time temperature, humidity and pressure values
- Second is when one observes the graph which is plotted in the web channel, he/she can easily understand the changes in the climatic conditions and also predict the future climatic conditions

Embedded System:

An embedded system is a special-purpose computer system designed to perform one or a few dedicated functions, sometimes with real-time computing constraints. It is usually embedded as part of a complete device including hardware and mechanical parts. In contrast, a general -purpose

computer, such as a personal computer, can do many different tasks depending on programming. Embedded systems have become very important today as they control many of the common devices we use.

Since the embedded system is dedicated to specific tasks, design engineers can optimize it, reducing the size and cost of the product, or increasing the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

Physically, embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, or the systems controlling nuclear power plants. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

In general, "embedded system" is not an exactly defined term, as many systems have some element of programmability. For example, Handheld computers share some elements with embedded systems such as the operating systems and microprocessors which power them but are not truly embedded systems, because they allow different applications to be loaded and peripherals to be connected.

An embedded system is some combination of computer hardware and software, either fixed in capability or programmable, that is specifically designed for a particular kind of application device. Industrial machines, automobiles, medical equipment, cameras, household appliances, airplanes, vending machines, and toys (as well as the more obvious cellular phone and PDA) are among the myriad possible hosts of an embedded system. Embedded systems that are programmable are provided with a programming interface, and embedded systems programming is a specialized occupation.

Certain operating systems or language platforms are tailored for the embedded market, such as Embedded Java and Windows XP Embedded. However, some low-end consumer products use very inexpensive microprocessors and limited storage, with the application and operating system both part of a single program. The program is written permanently into the system's memory in this case, rather than being loaded into RAM (Random Access Memory), as programs on a personal

computer.

In recent days, you are showered with variety of information about these embedded controllers in many places. All kinds of magazines and journals regularly dish out details about latest technologies, new devices; fast applications which make you believe that your basic survival is controlled by these embedded products. Now you can agree to the fact that these embedded products have successfully invaded into our world. You must be wondering about these embedded controllers or systems. The computer you use to compose your mails, or create a document or analyze the database is known as the standard desktop computer. These desktop computers are manufactured to serve many purposes and applications.

You need to install the relevant software to get the required processing facility. So, these desktop computers can do many things. In contrast, embedded controllers' carryout a specific work for which they are designed. Most of the time, engineers design these embedded controllers with a specific goal in mind. So, these controllers cannot be used in any other place.

Theoretically, an embedded controller is a combination of a piece of microprocessor-based hardware and the suitable software to undertake a specific task. These days designers have many choices in microprocessors/microcontrollers. Especially, in 8 bit and 32 bits, the available variety really may overwhelm even an experienced designer. Selecting a right microprocessor may turn out as a most difficult first step and it is getting complicated as new devices continue to pop-up very often.

Basic Blocks of Embedded System:

Now, the details of the various building blocks of the hardware of an embedded systems shown in Fig 1.1are

- Central Processing Unit (CPU)
- Memory (Read-only Memory and Random-Access Memory)
- Input devices
- Output devices
- Communication interfaces
- Application-specific circuitry

Central Processing Unit (CPU):

The Central Processing Unit (processor, in short) can be any of the following: microcontroller, microprocessor or Digital Signal Processor (DSP). A micro-controller is a low- cost processor. Its main attraction is that on the chip itself, there will be many other components such as memory, serial communication interface, analog-to digital converter etc. So, for small applications, a micro-controller is the best choice as the number of external components required will be very less. On the other hand, microprocessors are more powerful, but you need to use many external components with them. DSP is used mainly for applications in which signal processing is involved such as audio and video processing.

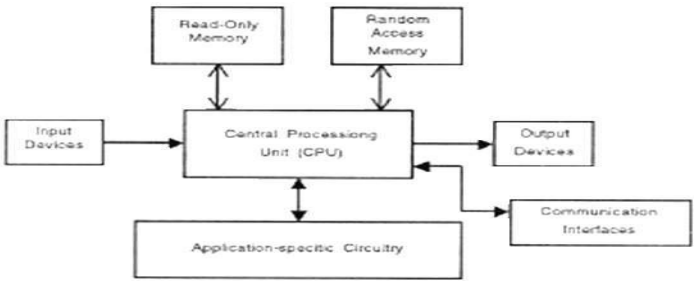


Figure.1.1: Block Diagram of Embedded System

Memory:

The memory is categorized as Random Access Memory (RAM) and Read Only Memory (ROM). The contents of the RAM will be erased if power is switched off to the chip, whereas ROM retains the contents even if the power is switched off. So, the firmware is stored in the ROM. When power is switched on, the processor reads the ROM; the program is executed.

Input devices:

Unlike the desktops, the input devices to an embedded system have very limited capability. There will be no keyboard or a mouse, and hence interacting with the embedded system is not an easy task. Many embedded systems will have a small keypad-you press one key to give a specific command. A keypad may be used to input only the digits. Many embedded systems used in process control do not have any input device for user interaction; they take inputs from sensors or transducers and produce electrical signals that are in turn fed to other

systems.

Output devices:

The output devices of the embedded systems also have very limited capability. Some embedded systems will have a few Light Emitting Diodes (LEDs) to indicate the health status of the system modules, or for visual indication of alarms. A small Liquid Crystal Display (LCD) may also be used to display some important parameters.

Communication Interfaces:

The embedded systems may need to, interact with other embedded systems at they may have to transmit data to a desktop. To facilitate this, the embedded systems are provided with one or a few communication interfaces such as RS232, RS422, RS485, Universal Serial Bus (USB), and IEEE 1394, Ethernet etc.

Application-specific circuitry:

Sensors, transducers, special processing and control circuitry may be required for an embedded system, depending on its application. This circuitry interacts with the processor to carry out the necessary work. The entire hardware has to be given power supply either through the 230 volts main supply or through a battery. The hardware has to design in such a way that the power consumption is minimized.

Security is the condition of being protected against danger or loss. In the general sense, security is a concept similar to safety. The nuance between the two is an added emphasis on being protected from dangers that originate from outside. Individuals or actions that encroach upon the condition of protection are responsible for the breach of security. The word "security" in general usage is synonymous with "safety," but as a technical term "security" means that something not only *is* secure but that it has been secured. One of the best options for providing good security is by using a technology named EMBEDDEDSYSTEM

Microcontroller:

In the Literature discussing microprocessors, we often see the term Embedded System. Microprocessors and Microcontrollers are widely used in embedded system products. An embedded system product uses a microprocessor (or Microcontroller) to do one task only. A printer is an example of embedded system since the processor inside it performs one task only, namely getting the data and printing it. Contrast this with a Pentium based PC can be used for any number of applications such as word processor, print-server, bank teller terminal, Video game, network server, or Internet terminal. Software for a variety of applications can be loaded and run. Of course, the reason a pc can perform myriad tasks is that it has RAM memory and an operating system that loads the application software into RAM memory and lets the CPU unit.

In an Embedded system, there is only one application software that is typically burned into ROM. An x86 PC contains or is connected to various embedded products such as keyboard, printer, modem, disk controller, sound card, CD-ROM drives, mouse, and so on. Each one of these peripherals has a Microcontroller inside it that performs only one task. For example, inside every mouse there is a Microcontroller to perform the task of finding the mouse position and sending it to the PC.

Application Areas:

Nearly 99 per cent of the processors manufactured end up in embedded systems. The embedded system market is one of the highest growth areas as these systems are used in very market segment- consumer electronics, office automation, industrial automation, biomedical engineering, wireless communication, data communication, telecommunications, transportation, military and so on.

Consumer appliances: At home we use a number of embedded systems which include digital camera, digital diary, DVD player, video recorders etc.

Office automation: The office automation products using embedded systems are copying machine, fax machine, key telephone, modem, printer, scanner etc

Microprocessors Vs Microcontrollers:

- Microprocessors are single-chip CPUs used in micro-computers.
- Microcontrollers and microprocessors are different in 3 main aspects: hardware architecture, applications, and instruction set features.
- Hardware architecture: A microprocessor is a single chip CPU while a microcontroller is a single IC contains a CPU and much of remaining circuitry of a complete computer (e.g., RAM, ROM, serial interface, parallel interface, timer, interrupt handling circuit).

Industrial automation: Today a lot of industries use embedded systems for process control. In hazardous industrial environment, where human presence has to be avoided, robots are used,

which are programmed to do specific jobs.

Medical electronics: Almost every medical equipment in the hospital is an embedded system. This equipment includes diagnostic aids such as ECG, EEG, blood pressure measuring devices, radiation, colonoscopy, endoscopy etc.

Computer networking: Computer networking products such as bridges, routers, Integrated Services Digital Networks (ISDN), Asynchronous Transfer Mode (ATM), X.25 and frame relay switches are embedded systems which implement the necessary data communication protocols

Telecommunications: In the field of telecommunications, the embedded systems can be categorized as subscriber terminals and network equipment. The subscriber terminals such as key telephones, ISDN phones, terminal adapters, web cameras are embedded systems. The network equipment includes multiplexers, multiple access systems, Packet Assemblers Dissemblers (PADs), sate11ite modems etc.

Security: Security of persons and information has always been a major issue. We need to protect our homes and offices; and also, the information we transmit and store. Developing embedded systems for devices at homes, offices, airports etc. for authentication and verification are embedded systems security applications is one of the most lucrative businesses nowadays.

• Applications: Microprocessors are commonly used as a CPU in computers while microcontrollers are found in small, minimum component designs performing control-oriented activities.

• Microprocessor instruction sets are processing Intensive.

• Their instructions operate on nibbles, bytes, words, or even doublewords.

• Addressing modes provide access to large arrays of data using pointers and offsets.

• They have instructions to set and clear individual bits and perform bit operations.

• They have instructions for input/output operations, event timing, enabling and setting priority levels for interrupts caused by external stimuli.

CHAPTER-2

EMBEDDED SYSTEM BASED DATA LOGGER SYSTEM

Introduction:

Weather or Climate is important part of human life Present innovations in technology mainly focus on controlling and monitoring of different activities. These are increasingly emerging to reach the human needs. Most of this technology is focused on efficient monitoring and controlling different activities.

The technology of technology discussed above is called IOT, IOT (Internet of Things) is viewed as an innovation and financial wave in the worldwide data industry after the Internet. The IoT is a wise system which associates all things to the Internet with the end goal of trading data and conveying through the data detecting gadgets as per concurred conventions. It accomplishes the objective of keen recognizing, finding, following, observing, and overseeing things. It is an augmentation and extension of Internet-based system, which grows the correspondence from human and human to human and things or things and things.

IoT frameworks help for the interaction between “things”. Also supports for more complex structures like distributed computing and development of distributed applications. Now a days most of IoT frameworks seem to focus on real-time data logging solutions. The data of the measured parameters are not useful if they are not transmitted fast and accurate manner to the users. Therefore, transmitted and processing the measured data is a very important aspect of the modern weather forecast. Transmission of the measured data could be done by a number of means: WI-FI link, GSM/GPRS link, satellite link direct, wired link, etc. Weather forecasting has to be reliable and accurate, regardless of its application

An efficient environmental monitoring system is required to monitor and assess the conditions in case of exceeding the prescribed level of parameters (e.g., noise, CO and radiation levels). When the objects like environment equipped with sensor devices, microcontroller and various software applications becomes a self-protecting and self-monitoring environment and it is also called as smart environment.

In such environment when some event occurs the alarm or LED alerts automatically. The effects due to the environmental changes on animals, plants and human beings can be monitored and controlled by smart environmental monitoring system. By using embedded intelligence into the environment makes the environment interactive with other objectives, this is one of the applications that smart environment targets.

Human needs demand different types of monitoring systems these are depends on the type of data gathered by the sensor devices. Event Detection based and Spatial Process Estimation are the two categories to which applications are classified. Initially the sensor devices are deployed in environment to detect the parameters (e.g. Temperature, Humidity, Pressure, LDR, noise, CO and radiation levels etc.) while the data acquisition, computation and controlling action (e.g., the

variations in the noise and CO levels with respect to the specified levels). Sensor devices are placed at different locations to collect the data to predict the behavior of a particular area of interest. The main aim of this paper is to design and implement an efficient monitoring system through which the required parameters are monitored remotely using internet and the data gathered from the sensors are stored in the cloud and to project the estimated trend on the web browser.

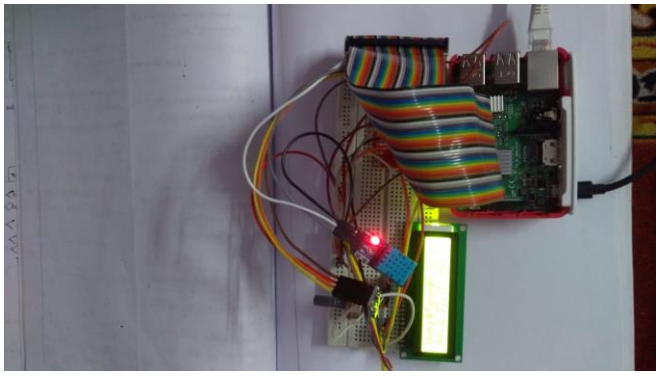
A solution for monitoring the noise and CO levels i.e., any parameter value crossing its threshold value ranges, for example CO levels in air in a particular area exceeding the normal levels etc., in the environment using wireless embedded computing system is proposed in this paper. The solution also provides an intelligent remote monitoring for a particular area of interest. In this paper we also present a trending results of collected or sensed data with respect to the normal or specified ranges of particular parameters. The embedded system is an integration of sensor devices, wireless communication which enables the user to remotely access the various parameters and store the data in cloud.

2.1 SYSTEM MODELLING

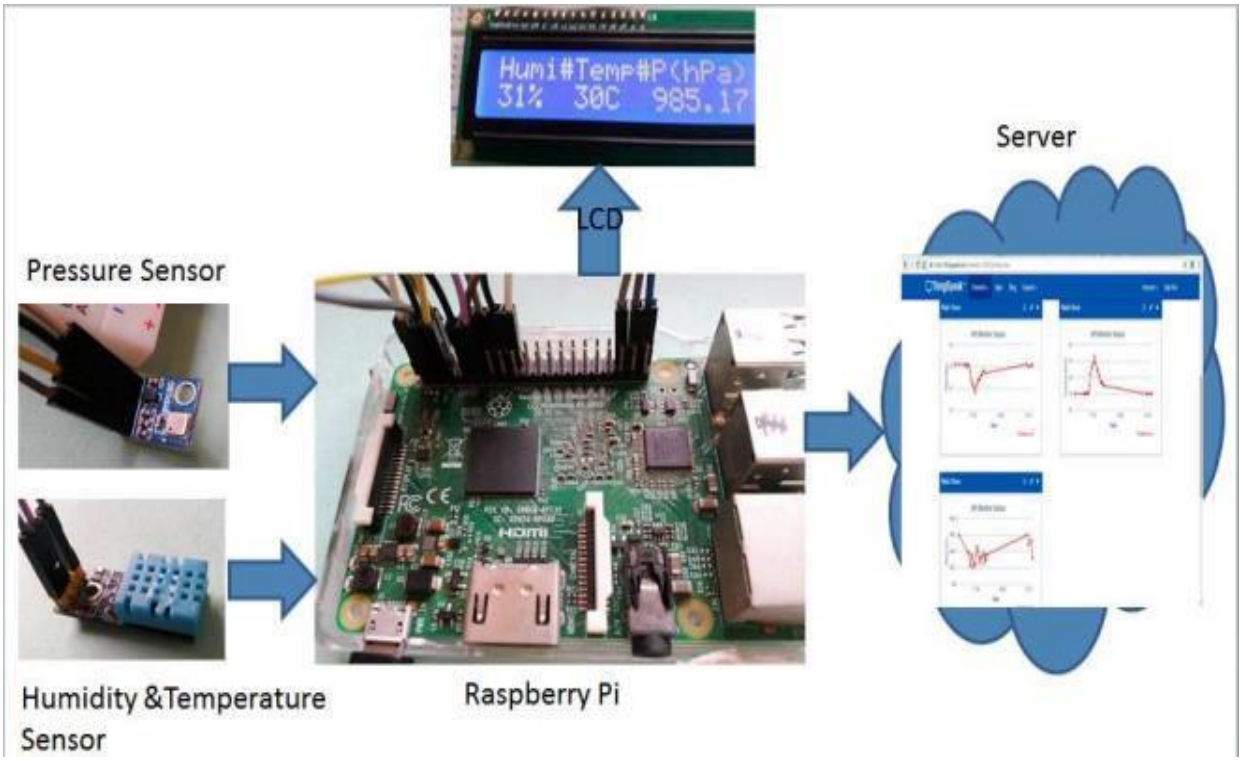
Proposed System will visualize and store various weather parameters such as temperature pressure and humidity as given above with the help of sensors interfaced to Raspberry.

The Raspberry pi will get all data, SD card on Pi stores the collected data as like memory card. Then at the output side LCD is to be connected for showing the result and on off relays for server access. To know the current weather status at remote location, the user can to log in on web browser by entering username and password given for particular server by the user.

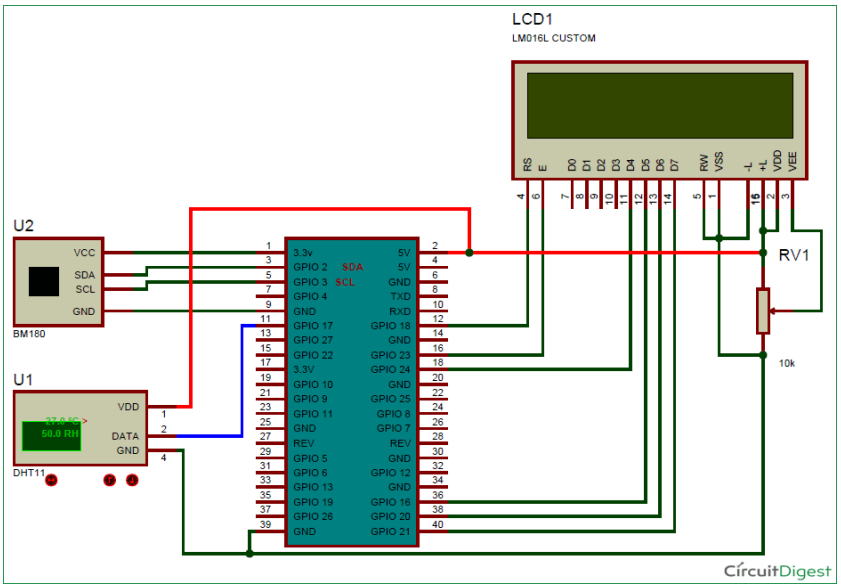
Web application opens after entering password and with the output graphical representation also obtain. Raspberry pi processed data will be updated continuously on cloud server & user will get to know the stored data on hourly and daily basis as shown in the below figure



- Humidity, Temperature and Pressure are three basic parameters to build any Weather Station and to measure environmental conditions. Here we are developing a weather station with Raspberry Pi.
- This IoT based Project aims to show the current Humidity, Temperature and Pressure parameters on the LCD as well on the Internet server using Raspberry Pi, which makes it a Raspberry Pi Weather Station. You can install this setup anywhere and can monitor the weather conditions of that place from anywhere in the world over the internet, it will not only show the current data but can also show the past values in the form of Graphs.
- We have used DHT11 Humidity & temperature sensor for sensing the temperature and BM180 Pressure sensor module for measuring barometric pressure. This Celsius scale Thermometer and percentage scale Humidity meter displays the ambient temperature and humidity through an LCD display and barometric pressure is displayed in millibar or hPa (hectopascal). All this data is sent to ThingSpeak server for live monitoring from anywhere in the world over internet.



The circuit diagram of the above system is as follows:

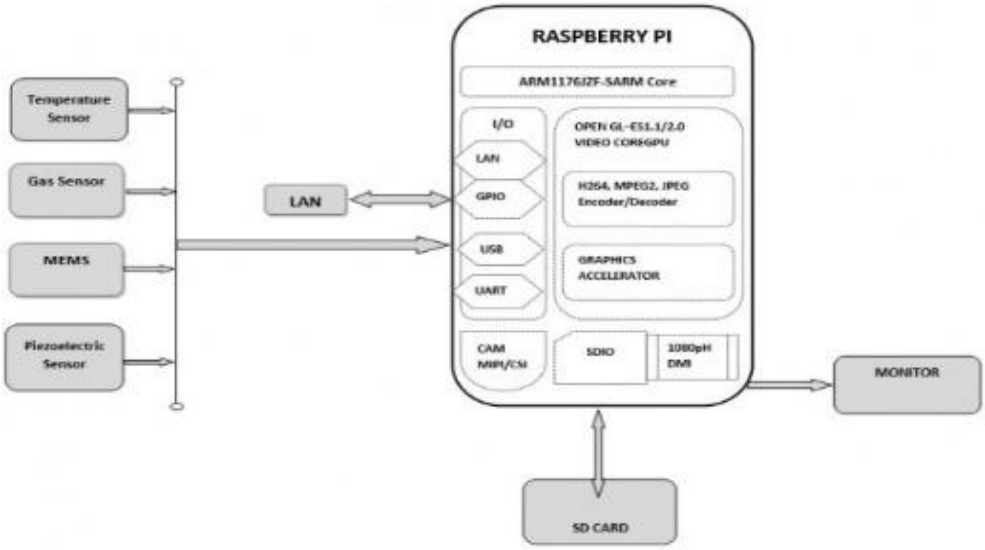


This IoT based project has four sections.

- The DHT11 sensor senses the Humidity & Temperature Data and BM180 sensor measures the atmospheric pressure.
- Raspberry Pi reads the DHT11 sensor module's output by using single wire protocol and BM180 pressure sensor's output by using I2C protocol and extracts both sensors values into a suitable number in percentage (humidity), Celsius scale (temperature), hectopascal or millibar (pressure).
- Then these values are sent to ThingSpeak server by using inbuilt Wi-Fi of Raspberry Pi 3.
- Finally, ThingSpeak analyses the data and shows it in a Graph form. A LCD is also used to display these values locally.

2.2 Block Diagram:

The block-diagram shown below represents the weather monitoring system, as discussed above the user will be able to observe and also analyse the real time weather parameters.



COMPONENTS USED IN THIS PROJECT:

Hardware Components:

Raspberry pi 3

DHT11

BMP180

LCD

Software Components:

Raspbian OS

ThingSpeak Web Server

Chapter-3

Raspberry PI

Introduction:

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote teaching of basic computer science in schools and in developing countries. The original model became far more popular than anticipated, selling outside its target market for uses such as robotics

3.1 Specifications:

3.1.1 Processor:

• Broadcom BCM2387 chipset.

• 1.2 GHz Quad-Core ARM Cortex-A53 (64Bit) with 512 KB shared L2 cache.

3.1.2 GPU

• Dual Core Video Core IV Multimedia Co-Processor. Provides Open GL ES 2.0, hardware-accelerated Open VG, and 1080p30 H.264 high-profile decode.

• Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure

3.1.3 Memory

• 1GB LPDDR2

3.1.4 Operating System

• Boots from Micro SD card, running a version of the Linux operating system or Windows 10 IoT

3.1.5 Dimensions

• 85 x 56 x 17mm

3.1.6 Power

• Micro USB socket 5V1, 2.5A

3.2 Connectors:

3.2.1 Ethernet

• 10/100 Base T Ethernet socket

3.2.2 Video Output

• HDMI (rev 1.3 & 1.4)

• Composite RCA (PAL and NTSC)

3.2.3 Audio Output

• Audio Output 3.5mm jack• HDMI• USB 4 x USB 2.0 Connector

3.2.4 GPIO Connector

• 40-pin 2.54 mm (100 mil) expansion header: 2x20 strip

• Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines

3.2.5 Camera Connector

• 15-pin MIPI Camera Serial Interface (CSI-2)

3.2.6 Display Connector

• Display Serial Interface (DSI) 15-way flat flex cable connector with two data lanes and a clock lane

3.2.7 Memory Card Slot

• Push/pull Micro SDIO

3.2.8 Graphic Processing Unit

The GPU provides Open GL ES 2.0, hardware-accelerated Open VG, and 1080p30 H.264 high-profile decode and is capable of 1Gpixel/s, 1.5Gtexel/s or 24 GFLOPs of general-purpose compute. It means that if you plug the Raspberry Pi 3 into your HDTV, you could watch Blu-ray quality video, using H.264 at 40MBits/s

3.3 Additional Features:

The biggest change that has been enacted with the Raspberry Pi 3 is an upgrade to a next generation main processor and improved connectivity with Bluetooth Low Energy (BLE) and BCM43143 Wi-Fi on board. Additionally, the Raspberry Pi 3 has improved power management, with an upgraded switched power source up to 2.5 Amps, to support more powerful external USB devices.

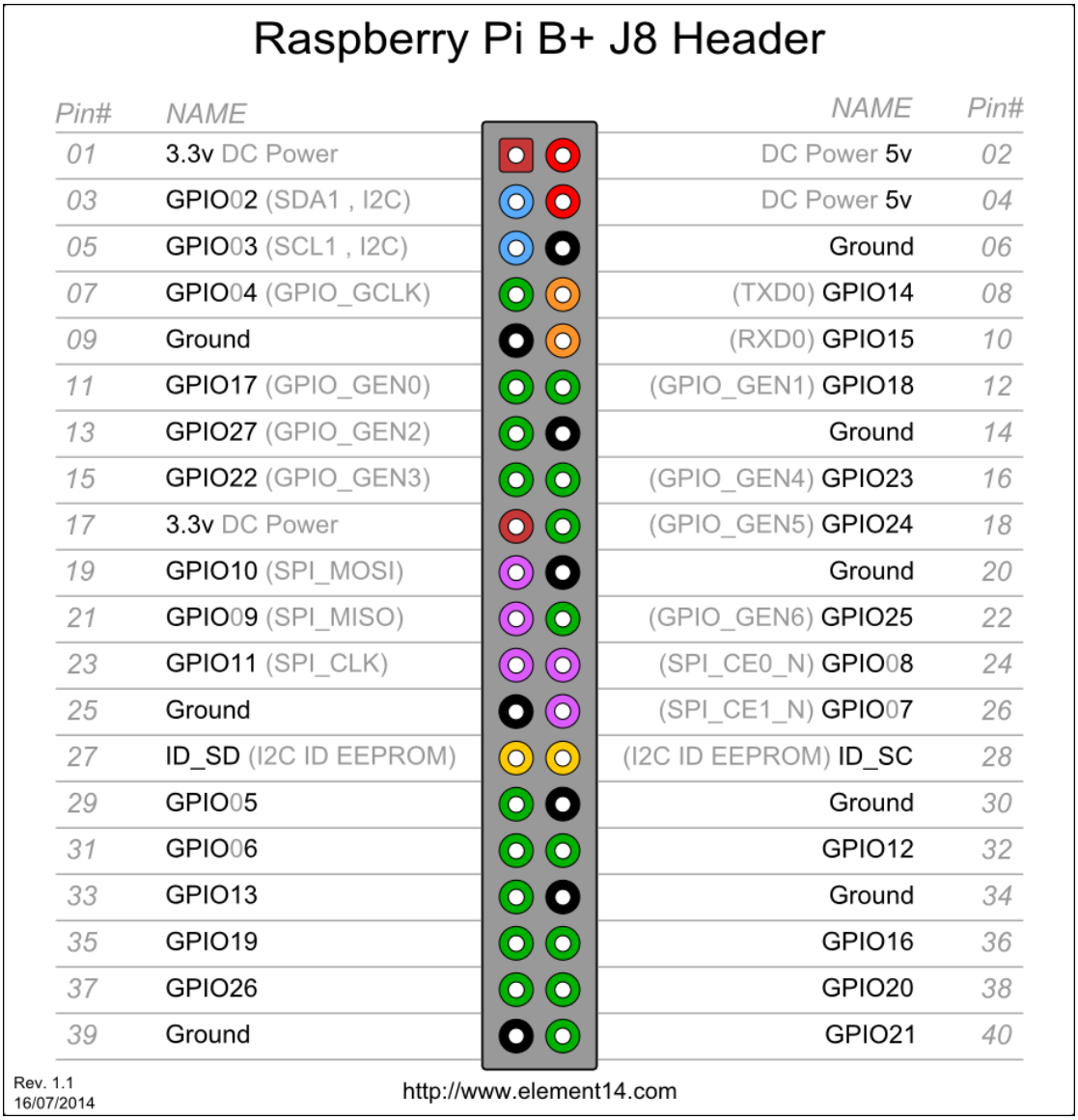
3.3.1 USB Ports

The Raspberry Pi 3's four built-in USB ports provide enough connectivity for a mouse, keyboard, or anything else that you feel the RPi needs, but if you want to add even more you can still use a USB hub. Keep in mind, it is recommended that you use a powered hub so as not to overtax the on-board voltage regulator. Powering the Raspberry Pi 3 is easy, just plug any USB power supply into the micro-USB port. There's no power button so the Pi will begin to boot as soon as power is applied, to turn it off simply remove power. The four built-in USB ports can even output up to 1.2A enabling you to connect more power-hungry USB devices (This does require a 2Amp micro USB Power Supply)

3.3.2 GPIO Pins

The low-level peripherals on the Pi make it great for hardware hacking. The 0.1" spaced 40-pin GPIO header on the Pi gives you access to 27 GPIO, UART, I2C, SPI as well as 3.3 and 5V sources. Each pin on the GPIO header is identical to its predecessor the Model B+.

GPIO The Raspberry Pi 3 features the same 40-pin general-purpose input-output (GPIO) header as all the Pins going back to the Model B+ and Model A+. Any existing GPIO hardware will work without modification; the only change is a switch to which UART is exposed on the GPIO's pins, but that's handled internally by the operating system.



Wedge Silk	Python (BCM)	WiringPi GPIO	Name	P1 Pin Number		Name	WiringPi GPIO	Python (BCM)	Wedge Silk
			3.3v DC Power	**1**	**2**	5v DC Power			
SDA		8	GPIO02 (SDA1, I2C)	**3**	**4**	5v DC Power			
SCL		9	GPIO03 (SCL1, I2C)	**5**	**6**	Ground			
G4	4	7	GPIO04 (GPIO_GCLK)	**7**	**8**	GPIO14 (TXD0)	15		TXO
			Ground	**9**	**10**	GPIO15 (RXD0)	16		RXI

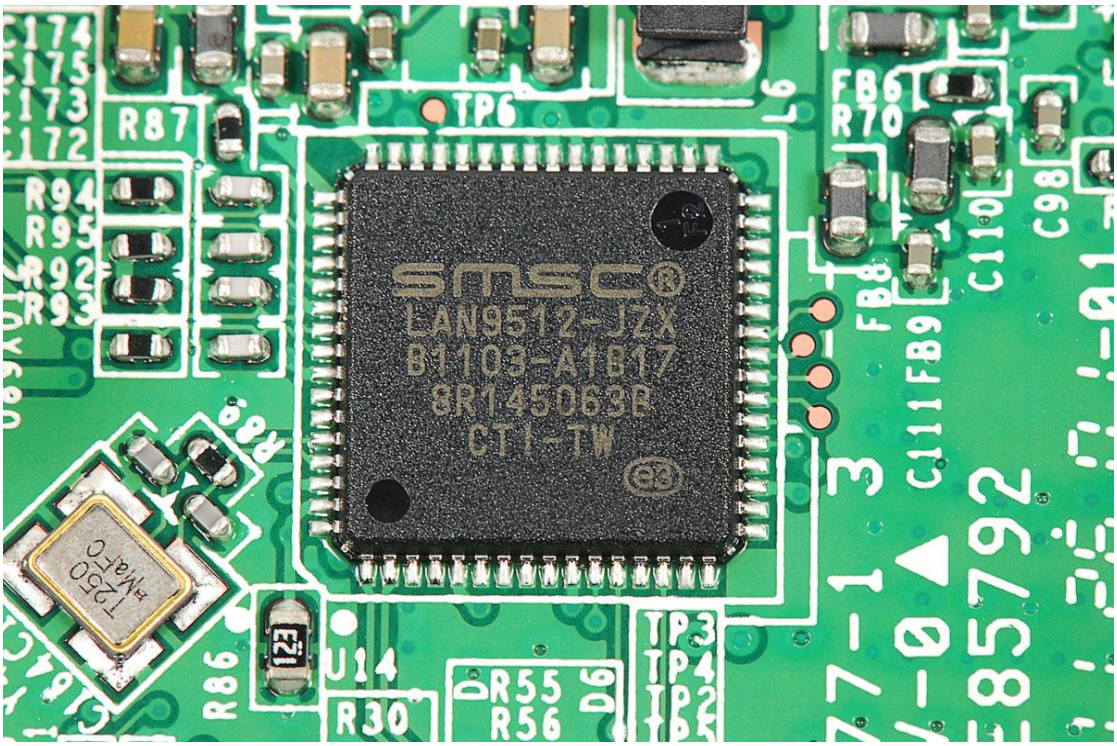
G17	17	0	GPIO17 (GPIO_GEN0)	**11**	**12**	GPIO18 (GPIO_GEN1)	1	18	G18
G27	27	2	GPIO27 (GPIO_GEN2)	**13**	**14**	Ground			
G22	22	3	GPIO22 (GPIO_GEN3)	**15**	**16**	GPIO23 (GPIO_GEN4)	4	23	G23
			3.3v DC Power	**17**	**18**	GPIO24 (GPIO_GEN5)	5	24	G24
MOSI		12	GPIO10 (SPI_MOSI)	**19**	**20**	Ground			
MISO		13	GPIO09 (SPI_MISO)	**21**	**22**	GPIO25 (GPIO_GEN6)	6	25	G25
CLK		(no worky 14)	GPIO11 (SPI_CLK)	**23**	**24**	GPIO08 (SPI_CE0_N)	10		CD0
			Ground	**25**	**26**	GPIO07 (SPI_CE1_N)	11		CE1
IDSD		30	ID_SD (I2C ID EEPROM)	**27**	**28**	ID_SC (I2C ID EEPROM)	31		IDSC
G05	5	21	GPIO05	**29**	**30**	Ground			
G6	6	22	GPIO06	**31**	**32**	GPIO12	26	12	G12
G13	13	23	GPIO13	**33**	**34**	Ground			
G19	19	24	GPIO19	**35**	**36**	GPIO16	27	16	G16
G26	26	25	GPIO26	**37**	**38**	GPIO20	28	20	G20
			Ground	**39**	**40**	GPIO21	29	21	G21

3.3.4 SoC

Built specifically for the new Pi 3, the Broadcom BCM2837 system-on-chip (SoC) includes four high-performance ARM Cortex-A53 processing cores running at 1.2GHz with 32kB Level 1 and 512kB Level 2 cache memory, a Video Core IV graphics processor, and is linked to a 1GB LPDDR2 memory module on the rear of the board.

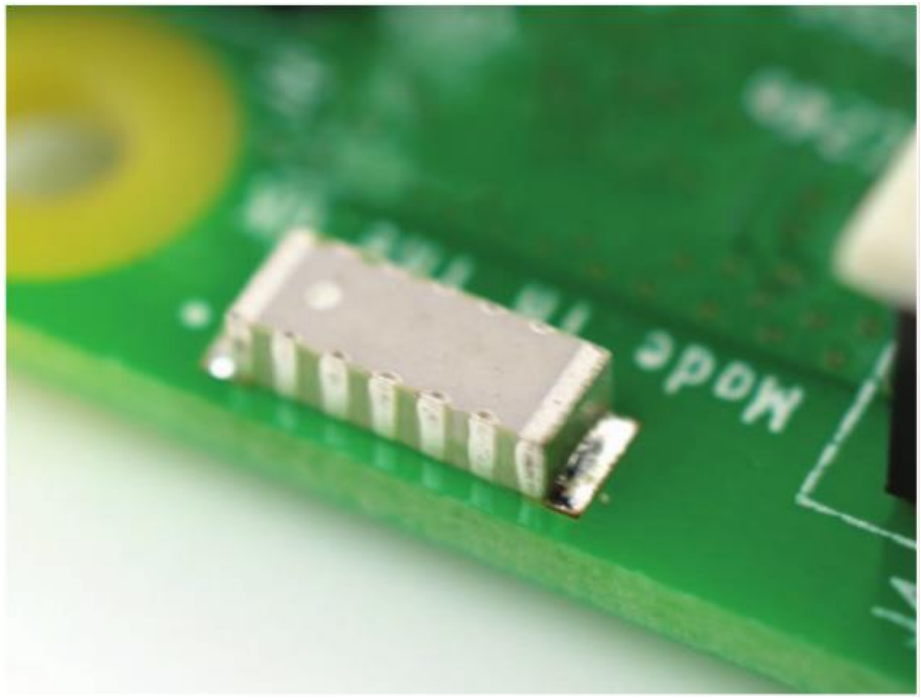
3.3.5 USB chip

The Raspberry Pi 3 shares the same SMSC LAN9514 chip as its predecessor, the Raspberry Pi 2, adding 10/100 Ethernet connectivity and four USB channels to the board. As before, the SMSC chip connects to the SoC via a single USB channel, acting as a USB-to-Ethernet adaptor and USB hub.



3.3.6 Antenna

There's no need to connect an external antenna to the Raspberry Pi 3. Its radios are connected to this chip antenna soldered directly to the board, in order to keep the size of the device to a minimum. Despite its diminutive stature, this antenna should be more than capable of picking up wireless LAN and Bluetooth signals – even through walls.



3.4 Key Improvements from Pi 2 Model B to Pi 3 Model B:

• Next Generation QUAD Core Broadcom BCM2837 64bit ARMv7 processor

• Processor speed has increased from 900MHz on Pi 2 to 1.25Ghz on the RPi 3 Model B

• BCM43143 Wi-Fi on board

• Bluetooth Low Energy (BLE) on board

• Upgraded switched power source up to 2.5 Amps (can now power even more powerful devices over USB ports)

The main differences are the quad core 64-bit CPU and on-board Wi-Fi and Bluetooth. The RAM remains 1GB and there is no change to the USB or Ethernet ports. However, the upgraded power management should mean the Pi 3 can make use of more power-hungry USB devices

For Raspberry Pi 3, Broadcom have supported us with a new SoC, BCM2837. This retains the same basic architecture as its predecessors BCM2835 and BCM2836, so all those projects and tutorials

which rely on the precise details of the Raspberry Pi hardware will continue to work. The 900MHz 32-bit quad-core ARMCortexA7 CPU complex has been replaced by a custom-hardened 1.2GHz 64-bit quad-core ARM Cortex-A53

In terms of size it is identical to the B+ and Pi 2. All the connectors and mounting holes are in the same place so all existing add-ons, HATs and cases should fit just fine although the power and activity LEDs have moved to make room for the Wi-Fi antenna.

The performance of the Pi 3 is roughly 50-60% faster than the Pi 2 which means it is ten times faster than the original Pi.

All of the connectors are in the same place and have the same functionality, and the board can still be run from a 5V micro-USB power adapter. This time round, we're recommending a 2.5A adapter if you want to connect power-hungry USB devices to the Raspberry Pi.

3.5 Applications:

- Desktop PC
- Wireless Printer
- Media Center
- Retro Gaming Machine
- Game server
- Robot Controller
- Web Server
- FM Radio Station
- Smart TV
- Home Auto Machine

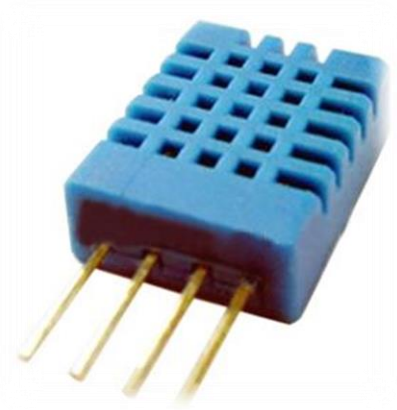
Chapter-4

DHT11 Humidity & Temperature Sensor

DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output.

Introduction:

DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal 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.



Each DHT11 element is strictly calibrated in the laboratory that is extremely accurate on humidity calibration. The calibration coefficients are stored as programmes in the OTP memory, which are used by the sensor's internal signal detecting process. The single-wire serial interface makes system integration quick and easy. Its small size, low power consumption and up-to-20-meter signal transmission making it the best choice for various applications, including those most demanding ones. The component is 4-pin single row pin package. It is convenient to connect and special packages can be provided according to users' request.

4.1. Technical Specifications:

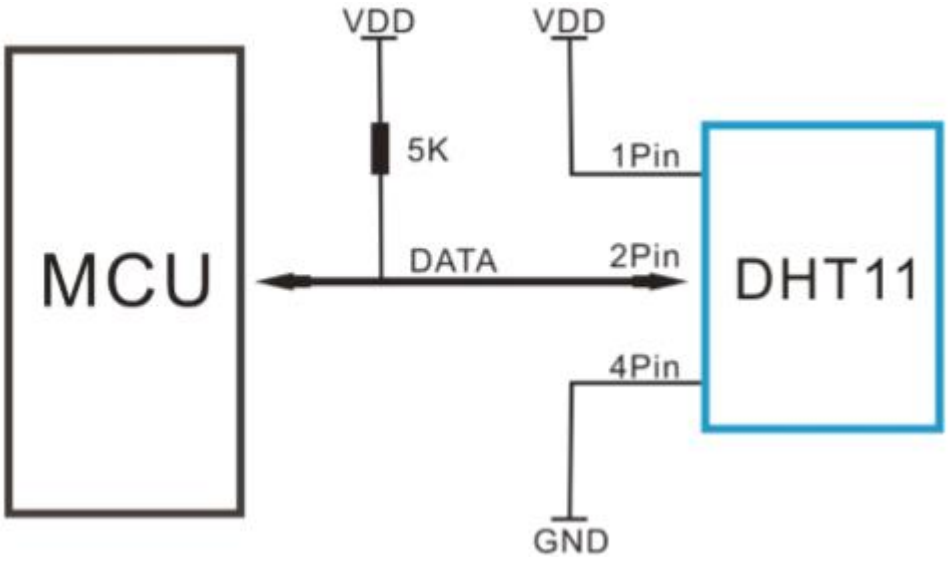
Overview:

Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

Detailed Specifications:

Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25℃		±4%RH	
	0-50℃			±5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0℃	30%RH		90%RH
	25℃	20%RH		90%RH
	50℃	20%RH		80%RH
Response Time (Seconds)	1/e(63%)25℃, 1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	
Long-Term Stability	Typical		±1%RH/year	
Temperature				
Resolution		1℃	1℃	1℃
		8 Bit	8 Bit	8 Bit
Repeatability			±1℃	
Accuracy		±1℃		±2℃
Measurement Range		0℃		50℃
Response Time (Seconds)	1/e(63%)	6 S		30 S

4.2. Typical Application:



Note: 3Pin – Null; MCU = Micro-computer Unite or single chip Computer

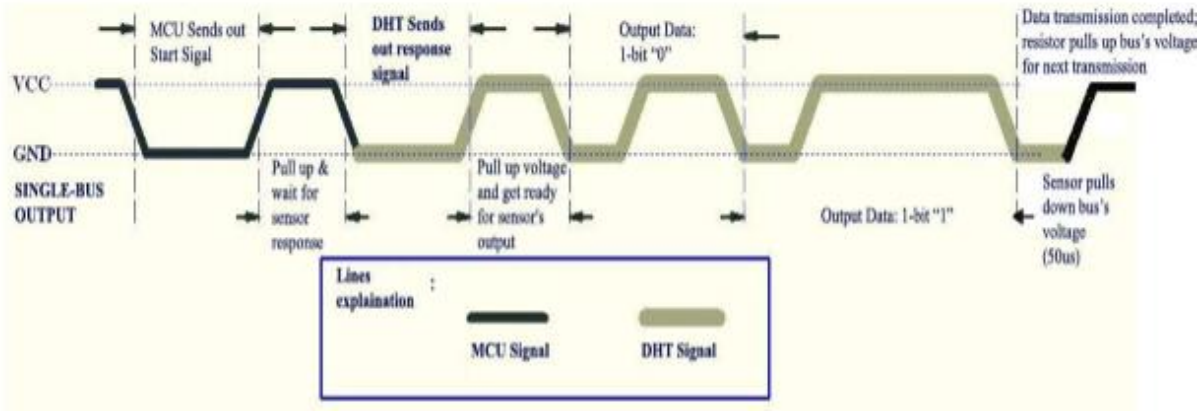
When the connecting cable is shorter than 20 metres, a 5K pull-up resistor is recommended; when the connecting cable is longer than 20 metres, choose an appropriate pull-up resistor as needed.

4. Power and Pin: DHT11's power supply is 3-5.5V DC. When power is supplied to the sensor, do not send any instruction to the sensor in within one second in order to pass the unstable status. One capacitor valued 100nF can be added between VDD and GND for power filtering.

5. Communication Process: Serial Interface (Single-Wire Two-Way) Single-bus data format is used for communication and synchronization between MCU and DHT11 sensor. One communication process is about 4ms.

Data consists of decimal and integral parts. A complete data transmission is 40bit, and the sensor sends higher data bit first. Data format: 8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit check sum. If the data transmission is right, the check-sum should be the last 8bit of "8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data".

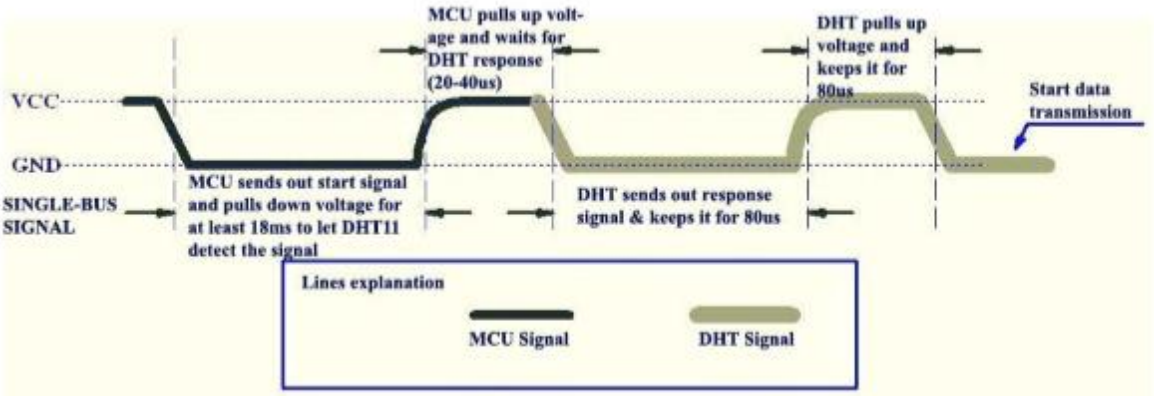
5.1 Overall Communication Process: When MCU sends a start signal, DHT11 changes from the low-power-consumption mode to the running-mode, waiting for MCU completing the start signal. Once it is completed, DHT11 sends a response signal of 40-bit data that include the relative humidity and temperature information to MCU. Users can choose to collect (read) some data. Without the start signal from MCU, DHT11 will not give the response signal to MCU. Once data is collected, DHT11 will change to the low power-consumption mode until it receives a start signal from MCU again.



Overall Communication Process

4.2.1 MCU Sends out Start Signal to DHT :

Data Single-bus free status is at high voltage level. When the communication between MCU and DHT11 begins, the programme of MCU will set Data Single-bus voltage level from high to low and this process must take at least 18ms to ensure DHT's detection of MCU's signal, then MCU will pull up voltage and wait 20-40us for DHT's response.



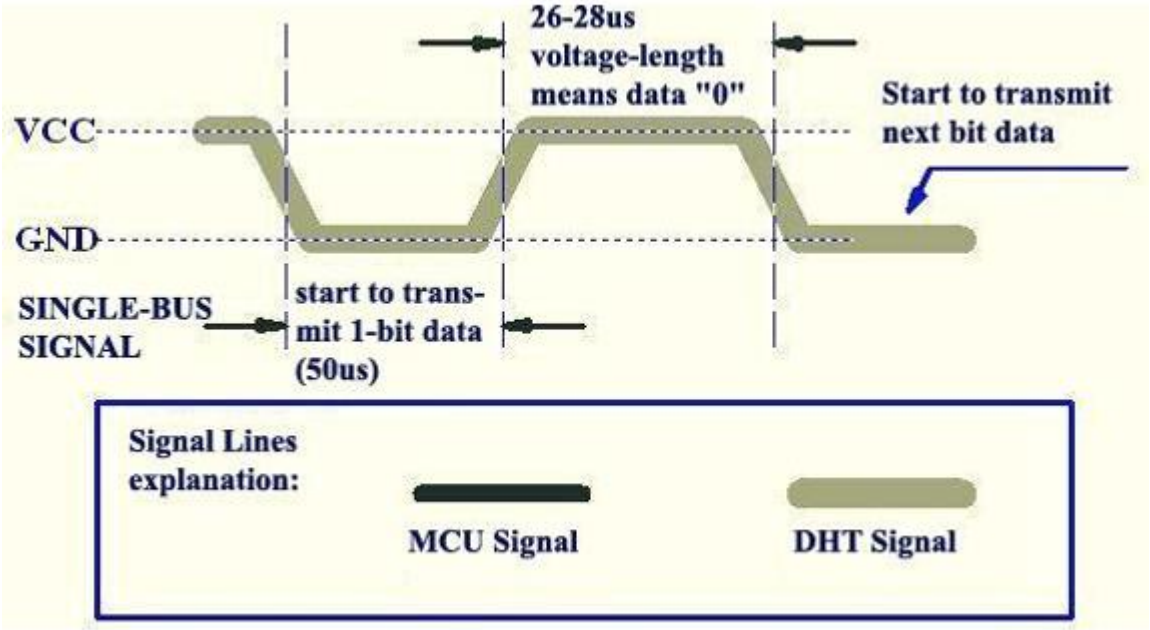
MCU Sends out Start Signal & DHT Responses

4.2.2 DHT Responses to MCU :

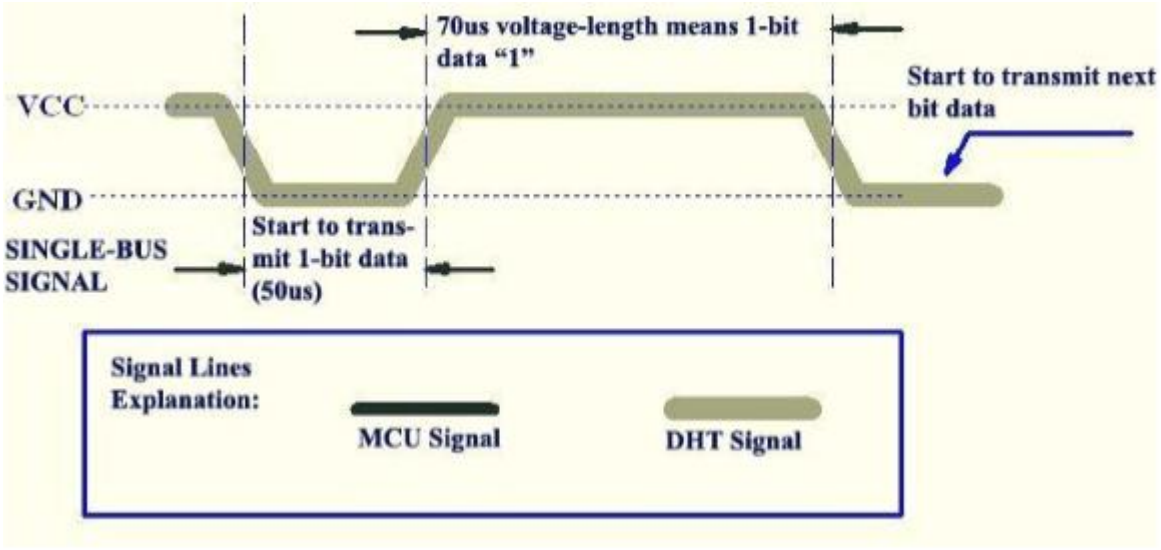
Once DHT detects the start signal, it will send out a low-voltage-level response signal, which lasts 80us. Then the programme of DHT sets Data Single-bus voltage level from low to high and keeps it for 80us for DHT's preparation for sending data.

When DATA Single-Bus is at the low voltage level, this means that DHT is sending the response signal. Once DHT sent out the response signal, it pulls up voltage and keeps it for 80us and prepares for data transmission.

When DHT is sending data to MCU, every bit of data begins with the 50us low-voltage-level and the length of the following high-voltage-level signal determines whether data bit is "0" or "1".



Data "0" Indication



Data "1" Indication

If the response signal from DHT is always at high-voltage-level, it suggests that DHT is not responding properly and please check the connection. When the last bit data is transmitted, DHT11 pulls down the voltage level and keeps it for 50us. Then the Single-Bus voltage will be pulled up by the resistor to set it back to the free status.

4.3 General Characteristics:

4.3.1Electrical Characteristics:

VDD=5V, T = 25℃

	Conditions	Minimum	Typical	Maximum
Power Supply	DC	3V	5V	5.5V
Current Supply	Measuring	0.5mA		2.5mA
	Average	0.2mA		1mA
	Standby	100uA		150uA
Sampling period	Second	1		

4.5. Attentions of application:

- Operating conditions Applying the DHT11 sensor beyond its working range stated in this datasheet can result in 3%RH signal shift/discrepancy. The DHT11 sensor can recover to the calibrated status gradually when it gets back to the normal operating condition and works within its range.

- Attention to chemical materials Vapor from chemical materials may interfere with DHT's sensitive-elements and debase its sensitivity. A high degree of chemical contaminant on can permanently damage the sensor.
- Restoration process when 1 & 2 happen Step one: Keep the DHT sensor at the condition of Temperature 50~60Celsius, humidity <10%RH for 2 hours; Step two: K keep the DHT sensor at the condition of Temperature 20~30Celsius, humidity >70%RH for 5 hours.

- Light etc. Long me exposure to strong sunlight and ultraviolet may debase DHT's performance.
- Connection wires: The quality of connection wires will affect the quality and distance of communication and high-quality shielding-wire is recommended.
- Other attentions * Welding temperature should be below 260Celsius and contact should take less than 10 seconds. * Avoid using the sensor under dew condition. * Do not use this product in safety or emergency stop devices or any other occasion that failure of DHT11 may cause personal injury. * Storage: Keep the sensor at temperature 10-40℃, humidity <60%RH.

Chapter-5

BMP180 DIGITAL PRESSURE SENSOR

Introduction:

The BMP180 is the function compatible successor of the BMP085, a new generation of high precision digital pressure sensors for consumer applications.

The ultra-low power, low voltage electronics of the BMP180 is optimized for use in mobile phones, PDAs, GPS navigation devices and outdoor equipment. With a low altitude noise of merely 0.25m at fast conversion time, the BMP180 offers superior performance. The I2C interface allows for easy system integration with a microcontroller.

5.1 General Characteristics of BMP 180:

The BMP180 is based on piezo-resistive technology for EMC robustness, high accuracy and linearity as well as long term stability.

5.1.1 Electrical characteristics:

If not stated otherwise, the given values are ±3-Sigma values over temperature/voltage range in the given operation mode. All values represent the new parts specification; additional solder drift is shown separately.

Operating conditions, output signal and mechanical characteristics

Parameter	Symbol	Condition	Min	Typ	Max	Units
Operating temperature	T_A	operational	-40		+85	°C
		full accuracy	0		+65	
Supply voltage	V_{DD}	ripple max. 50mVpp	1.8	2.5	3.6	V
			1.62	2.5	3.6	
Supply current @ 1 sample / sec. 25°C	I_{DDLOW}	ultra low power mode		3		µA
	I_{DDSTD}	standard mode		5		µA
	I_{DDHR}	high resolution mode		7		µA
	I_{DDUHR}	Ultra high res. mode		12		µA
	I_{DDAR}	Advanced res. mode		32		µA
Peak current	I_{peak}	during conversion		650	1000	µA
Standby current	I_{DDSBM}	@ 25°C		0.1	4[1]	µA
Relative accuracy pressure V_{DD} = 3.3V		950 ... 1050 hPa @ 25 °C		±0.12		hPa
				±1.0		m
		700 ... 900hPa 25 ... 40 °C		±0.12		hPa
				±1.0		m
Absolute accuracy pressure V_{DD} = 3.3V		300 ... 1100 hPa 0 ... +65 °C	-4.0	-1.0*	+2.0	hPa
		300 ... 1100 hPa -20 ... 0 °C	-6.0	-1.0*	+4.5	hPa
Resolution of output data		pressure		0.01		hPa
		temperature		0.1		°C
Noise in pressure		see table on page 12-13				
Absolute accuracy temperature V_{DD} = 3.3V		@ 25 °C	-1.5	±0.5	+1.5	°C
		0 ... +65 °C	-2.0	±1.0	+2.0	°C

* Long term stability is specified in the full accuracy operating pressure range 0 … 65°C

5.1.2. Absolute maximum ratings:

Parameter	Condition	Min	Max	Units
Storage temperature		-40	+85	°C
Supply voltage	all pins	-0.3	+4.25	V
ESD rating	HBM, R = 1.5kΩ, C = 100pF		±2	kV
Overpressure			10,000	hPa

The BMP180 has to be handled as Electrostatic Sensitive Device (ESD).

5.2. Operation

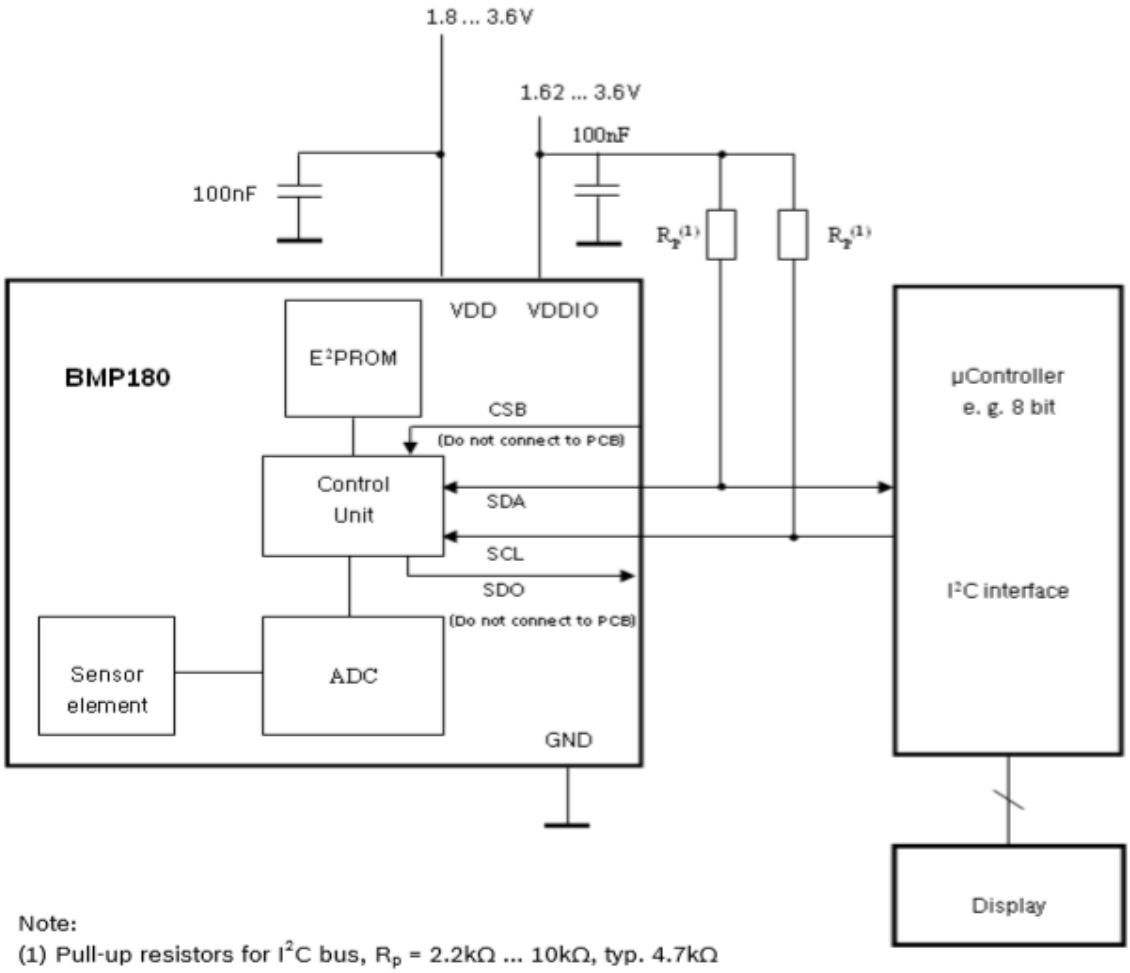
5.2.1 General description:

The BMP180 is designed to be connected directly to a microcontroller of a mobile device via the I2C bus. The pressure and temperature data have to be compensated by the calibration data of the E2PROM of the BMP180.

5.2.2 General function and application schematics:

The BMP180 consists of a piezo-resistive sensor, an analog to digital converter and a control unit with E2PROM and a serial I2C interface. The BMP180 delivers the uncompensated value of pressure and temperature. The E2PROM has stored 176 bit of individual calibration data. This is used to compensate offset, temperature dependence and other parameters of the sensor.

- UP = pressure data (16 to 19 bit)
- UT = temperature data (16 bit)



Typical application circuit

5.2.3 Measurement of pressure and temperature:

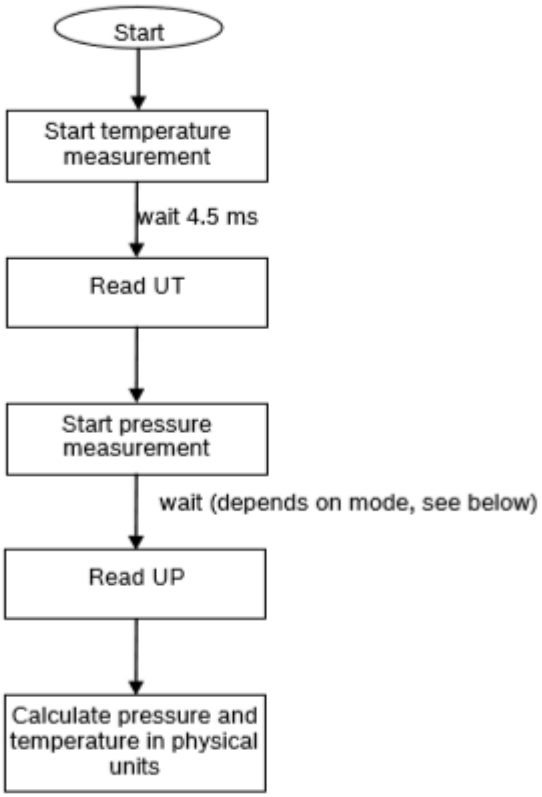
For all calculations presented here an ANSI C code is available from Bosch Sensor Tec (“BMP180 _API”).

The microcontroller sends a start sequence to start a pressure or temperature measurement. After converting time, the result value (UP or UT, respectively) can be read via the I2C interface. For calculating temperature in °C and pressure in hPa, the calibration data has to be used. These constants can be read out from the BMP180 E2PROM via the I2C interface at software initialization.

The sampling rate can be increased up to 128 samples per second (standard mode) for dynamic measurement. In this case, it is sufficient to measure the temperature only once per second and to use this value for all pressure measurements during the same period.

The mode (ultra-low power, standard, high, ultra-high resolution) can be selected by the variable oversampling setting (0, 1, 2, 3) in the C code.

Calculation of true temperature and pressure in steps of 1Pa (= 0.01hPa = 0.01mbar) and temperature in steps of 0.1°C



Measurement flow BMP180

5.2.4 Hardware pressure sampling accuracy modes:

By using different modes, the optimum compromise between power consumption, speed and resolution can be selected, see below table.

Overview of BMP180 hardware accuracy modes, selected by driver software via the variable oversampling setting:

Mode	Parameter *oversampling_setting*	Internal number of samples	Conversion time pressure max. [ms]	Avg. current @ 1 sample/s typ. [µA]	RMS noise typ. [hPa]	RMS noise typ. [m]
ultra low power	0	1	4.5	3	0.06	0.5
standard	1	2	7.5	5	0.05	0.4
high resolution	2	4	13.5	7	0.04	0.3
ultra high resolution	3	8	25.5	12	0.03	0.25

For further information on noise characteristics see the relevant application note “Noise in pressure sensor applications”.

All modes can be performed at higher speeds, e.g. up to 128 times per second for standard mode, with the current consumption increasing proportionally to the sample rate.

5.2.5. Software pressure sampling accuracy modes:

For applications where a low noise level is critical, averaging is recommended if the lower bandwidth is acceptable. Oversampling can be enabled using the software API driver (with OSR = 3).

Overview of BMP180 software accuracy mode, selected by driver software via the variable software oversampling setting

Mode	Parameter *oversampling_setting*	software_ oversampling_setting	Conversion time pressure max. [ms]	Avg. current @ 1 sample/s typ. [µA]	RMS noise typ. [hPa]	RMS noise typ. [m]
Advanced resolution	3	1	76.5	32	0.02	0.17

5.2.6 Calibration coefficients:

The 176-bit E2PROM is partitioned in 11 words of 16 bit each. These contain 11 calibration coefficients. Every sensor module has individual coefficients. Before the first calculation of temperature and pressure, the master reads out the E2PROM data. The data communication can be checked by checking that none of the words has the value 0 or 0xFFFF.

Calibration coefficients

	BMP180 reg adr	
Parameter	MSB	LSB
AC1	0xAA	0xAB
AC2	0xAC	0xAD
AC3	0xAE	0xAF
AC4	0xB0	0xB1
AC5	0xB2	0xB3
AC6	0xB4	0xB5
B1	0xB6	0xB7
B2	0xB8	0xB9
MB	0xBA	0xBB
MC	0xBC	0xBD
MD	0xBE	0xBF

5.2.7 Calculating absolute altitude:

With the measured pressure p and the pressure at sea level p0 e.g. 1013.25hPa, the altitude in meters can be calculated with the international barometric formula:

$$\text{altitude} = 44330 * \left(1 - \left(\frac{p}{p_0}\right)^{\frac{1}{5.255}}\right)$$

Thus, a pressure change of Δp = 1hPa corresponds to 8.43m at sea level.

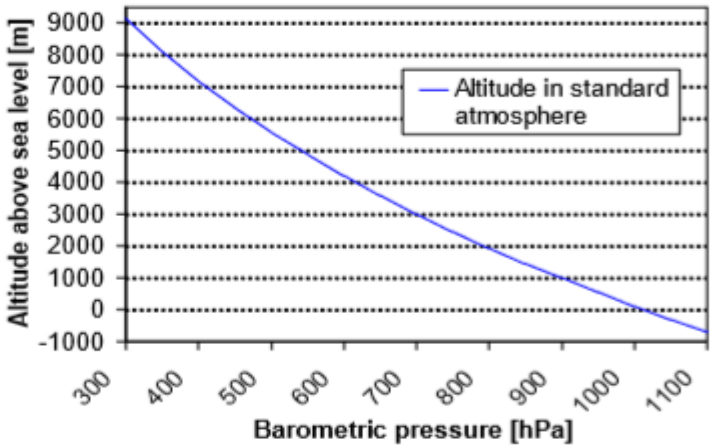


Figure 5: Transfer function: Altitude over sea level – Barometric pressure

3.7 Calculating pressure at sea level

With the measured pressure *p* and the absolute altitude the pressure at sea level can be calculated:

$$p_0 = \frac{p}{\left(1 - \frac{\text{altitude}}{44330}\right)^{5.255}}$$

Thus, a difference in altitude of Δaltitude = 10m corresponds to 1.2hPa pressure change at sea level.

5.3. Global Memory Map:

The memory map below shows all externally accessible data registers which are needed to operate BMP180. The left columns show the memory addresses. The columns in the middle depict the content of each register bit. The colours of the bits indicate whether they are read-only, write-only or read- and writable. The memory is volatile so that the writable content has to be re-written after each

power-on. Not all register addresses are shown. These registers are reserved for further Bosch factory testing and trimming.

Register Name	Register Adress	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Reset state
out_xlsb	F8h	adc_out_xlsb<7:3>					0	0	0	00h
out_lsb	F7h	adc_out_lsb<7:0>								00h
out_msb	F6h	adc_out_msb<7:0>								80h
ctrl_meas	F4h	oss<1:0>		sco	measurement control					00h
soft reset	E0h	reset								00h
id	D0h	id<7:0>								55h
calib21 downto calib0	BFh down to AAh	calib21<7:0> down to calib0<7:0>								n/a

Registers:	Control registers	Calibration registers	Data registers	Fixed
Type:	read / write	read only	read only	read only

Memory map

Measurement control (register F4h <4:0>): Controls measurements.

SCO (register F4h <5>): Start of conversion. The value of this bit stays “1” during conversion and is reset to “0” after conversion is complete (data registers are filled).

Oss (register F4h <7:6>): controls the oversampling ratio of the pressure measurement (00b: single, 01b: 2 times, 10b: 4 times, 11b: 8 times).

Soft reset (register E0h): Write only register. If set to 0xB6, will perform the same sequence as power on reset.

Chip-id (register D0h): This value is fixed to 0x55 and can be used to check whether communication is functioning.

After conversion, data registers can be read out in any sequence (i.e. MSB first or LSB first). Using a burst read is not mandatory.

5.3.1 I2C Interface:

I2C is a digital two wire interface

- Clock frequencies up to 3.4Mbit/sec. (I2C standard, fast and high-speed mode supported)
- SCL and SDA needs a pull-up resistor, typ. 4.7kOhm to VDDIO (one resistor each for all the I2C bus)
- The I2C bus is used to control the sensor, to read calibration data from the E2PROM and to read the measurement data when A/D conversion is finished. SDA (serial data) and SCL (serial clock) have open-drain outputs

5.3.2 Device and register address:

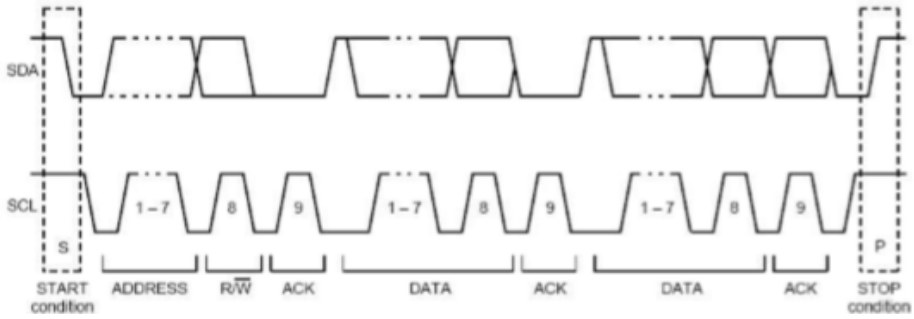
The BMP180 module address is shown below. The LSB of the device address distinguishes between read (1) and write (0) operation, corresponding to address 0xEF (read) and 0xEE (write).

BMP180 addresses.

A7	A6	A5	A4	A3	A2	A1	W/R
1	1	1	0	1	1	1	0/1

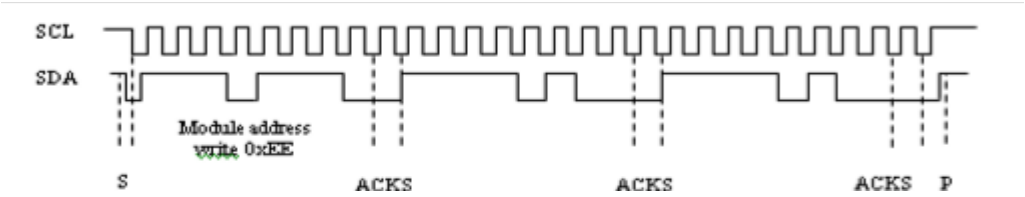
5.3 I²C protocol

The I²C interface protocol has special bus signal conditions. Start (S), stop (P) and binary data conditions are shown below. At start condition, SCL is high and SDA has a falling edge. Then the slave address is sent. After the 7 address bits, the direction control bit R/W selects the read or write operation. When a slave device recognizes that it is being addressed, it should acknowledge by pulling SDA low in the ninth SCL (ACK) cycle.
At stop condition, SCL is also high, but SDA has a rising edge. Data must be held stable at SDA when SCL is high. Data can change value at SDA only when SCL is low.
Even though V_{DDIO} can be powered on before V_{DD}, there is a chance of excessive power consumption (a few mA) if this sequence is used, and the state of the output pins is undefined so that the bus can be locked. Therefore, V_{DD} *must* be powered before V_{DDIO} unless the limitations above are understood and not critical.



I2C protocol

5.3.3 Start temperature and pressure measurement: The timing diagrams to start the measurement of the temperature value UT and pressure value UP are shown below. After start condition the master sends the device address write, the register address and the control register data. The BMP180 sends an acknowledgement (ACKS) every 8 data bits when data is received. The master sends a stop condition after the last ACKS.



Timing diagram for starting pressure measurement

Abbreviations:

S Start

P Stop

ACKS Acknowledge by Slave

ACKM Acknowledge by Master

NACKM Not Acknowledge by Master

Control registers values for different internal oversampling setting (oss)

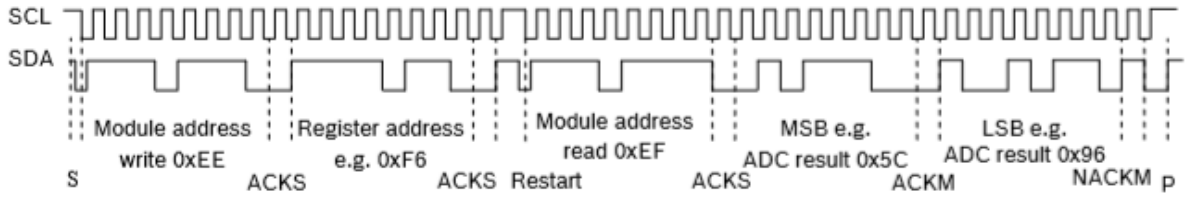
Measurement	Control register value (register address 0xF4)	Max. conversion time [ms]
Temperature	0x2E	4.5
Pressure (oss = 0)	0x34	4.5
Pressure (oss = 1)	0x74	7.5
Pressure (oss = 2)	0xB4	13.5
Pressure (oss = 3)	0xF4	25.5

5.3.3.1 Read A/D conversion result or E2PROM data: To read out the temperature data word UT (16 bit), the pressure data word UP (16 to 19 bit) and the E2PROM data proceed as follows:

After the start condition the master sends the module address write command and register address. The register address selects the read register:

E2PROM data registers 0xAA to 0xBF Temperature or pressure value UT or UP 0xF6 (MSB), 0xF7 (LSB), optionally 0xF8 (XLSB)

Then the master sends a restart condition followed by the module address read that will be acknowledged by the BMP180 (ACKS). The BMP180 sends first the 8 MSB, acknowledged by the master (ACKM), then the 8 LSB. The master sends a "not acknowledge" (NACKM) and finally a stop condition.

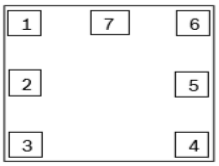


Timing diagram read 16-bit A/D conversion result

5.4 Package

5.4.1Pin configuration

Picture shows the device in top view. Device pins are shown here transparently only for orientation purposes.



Layout pin configuration BMP180

Pin No- Name- Function

1 -CSB* -Chip select

2 -VDD -Power supply

3 -VDDIO- Digital power supply

4- SDO*- SPI output

5- SCL I2-C serial bus clock input

6- SDA- I2C serial bus data (or SPI input)

7- GND- Ground

* A pin compatible product variant with SPI interface is possible upon customer's request. For I2C (standard case) CSB and SDO are not used, they have to be left open. All pins have to be soldered to the PCB for symmetrical stress input even though they are not connected internally.

5.5 Moisture sensitivity level and soldering:

The BMP180 is classified MSL 1 (moisture sensitivity level) according to IPC/JEDEC standards JSTD-020D and J-STD-033A.

The device can be soldered Pb-free with a peak temperature of 260°C for 20 to 40 sec. The minimum height of the solder after reflow shall be at least 50µm. This is required for good mechanical decoupling between the sensor device and the printed circuit board (PCB).

To ensure good solder-ability, the devices shall be stored at room temperature (20°C).

The soldering process can lead to an offset shift.

5.6 RoHS compliancy:

The BMP180 sensor meets the requirements of the EC directive "Restriction of hazardous substances (RoHS)".

The BMP180 sensor is also halogen-free.

5.6.1 Mounting and assembly recommendations:

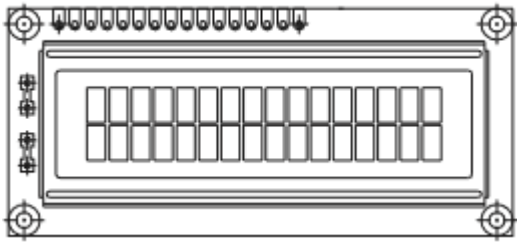
- In order to achieve the specified performance for you design, the following recommendations and the “Handling, soldering & mounting instructions BMP180” should be taken into consideration when mounting a pressure sensor on a printed-circuit board (PCB):
- The clearance above the metal lid shall be 0.1mm at minimum.
- For the device housing appropriate venting needs to be provided in case the ambient pressure shall be measured.
- Liquids shall not come into direct contact with the device.
- During operation the sensor is sensitive to light, which can influence the accuracy of the measurement (photo-current of silicon).
- The BMP180 shall not the placed close the fast heating parts. In case of gradients > 3°C/sec. it is recommended to follow Bosch Sensor Tec application note ANP015, "Correction of errors induced by fast temperature changes".

5.7 Applications:

- Indoor navigation
- GPS-enhancement for dead-reckoning, slope detection, etc.
- Sport devices, e.g. altitude profile
- Weather forecast
- Vertical velocity indication (rise/sink speed)

Chapter -6

16 x 2 Character LCD



Introduction:

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. ... A 16x2 LCD means it can display 16 characters per line and there are 2 such lines.

6.1 Features:

- 5 x 8 dots with cursor
- Built-in controller (KS 0066 or Equivalent)
- + 5V power supply (Also available for + 3V)
- 1/16 duty cycle
- B/L to be driven by pin 1, pin 2 or pin 15, pin 16 or A.K (LED)
- N.V. optional for + 3V power supply

DISPLAY CHARACTER ADDRESS CODE:

Display Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DD RAM Address	00	01														0F
DD RAM Address	40	41														4F

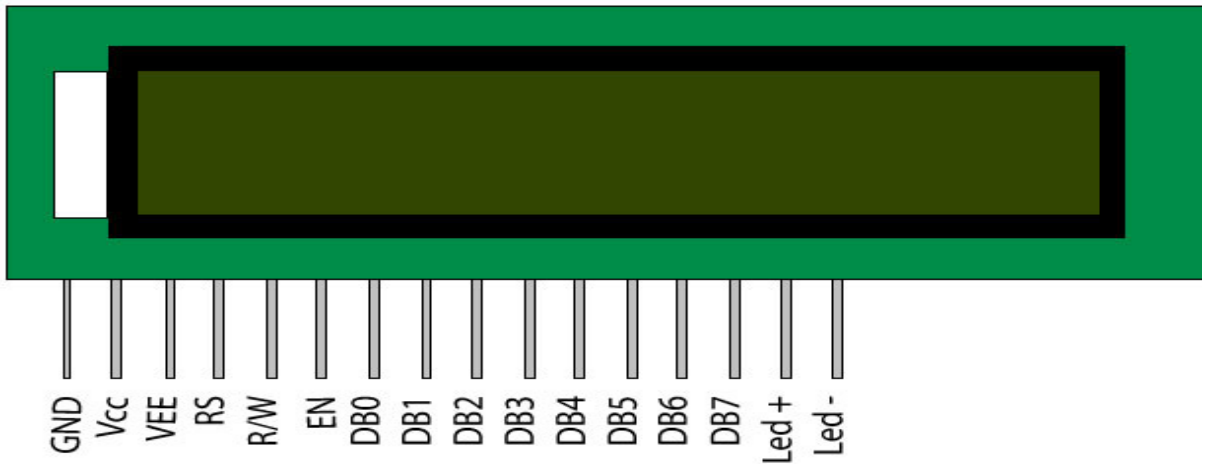
MECHANICAL DATA		
ITEM	STANDARD VALUE	UNIT
Module Dimension	80.0 x 36.0	mm
Viewing Area	66.0 x 16.0	mm
Dot Size	0.56 x 0.66	mm
Character Size	2.96 x 5.56	mm

ABSOLUTE MAXIMUM RATING					
ITEM	SYMBOL	STANDARD VALUE			UNIT
		MIN.	TYP.	MAX.	
Power Supply	VDD-VSS	- 0.3	–	7.0	V
Input Voltage	VI	- 0.3	–	VDD	V

NOTE: VSS = 0 Volt, VDD = 5.0 Volt

ELECTRICAL SPECIFICATIONS							
ITEM	SYMBOL	CONDITION		STANDARD VALUE			UNIT
				MIN.	TYP.	MAX.	
Input Voltage	VDD	VDD = + 5V		4.7	5.0	5.3	V
		VDD = + 3V		2.7	3.0	5.3	V
Supply Current	IDD	VDD = 5V		–	1.2	3.0	mA
Recommended LC Driving Voltage for Normal Temp. Version Module	VDD - V0	- 20 °C		–	–	–	V
		0°C		4.2	4.8	5.1	
		25°C		3.8	4.2	4.6	
		50°C		3.6	4.0	4.4	
		70°C		–	–	–	
LED Forward Voltage	VF	25°C		–	4.2	4.6	V
LED Forward Current	IF	25°C	Array	–	130	260	mA
			Edge	–	20	40	
EL Power Supply Current	IEL	Vel = 110VAC:400Hz		–	–	5.0	mA

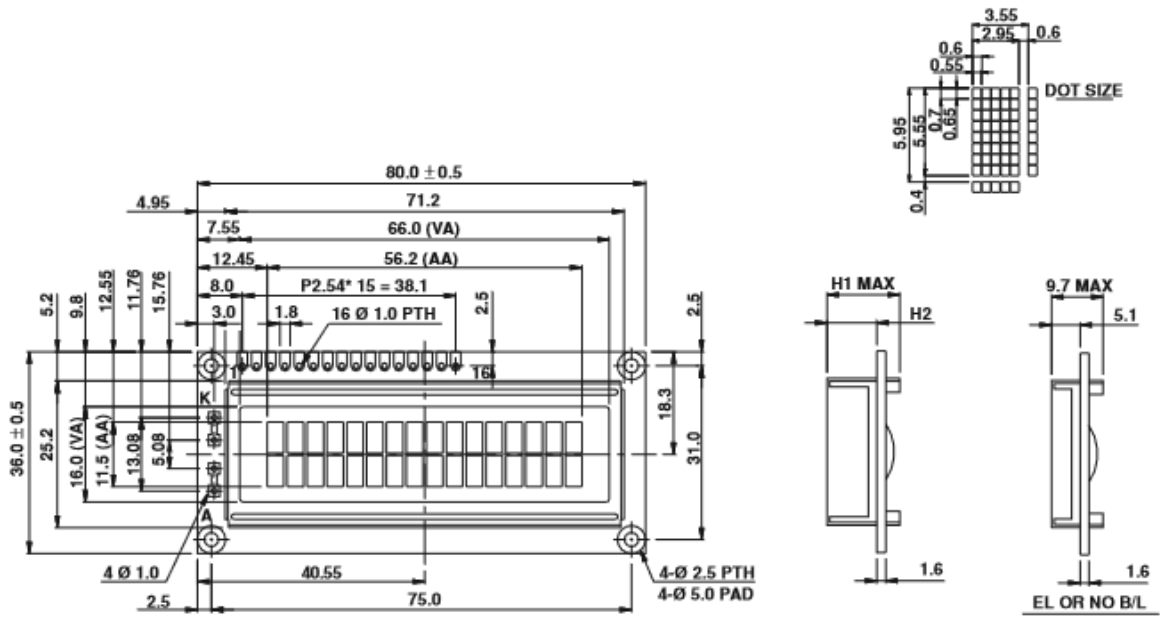
16 x 2 Character LCD



6.2.PIN DESCRIPTION:

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	V_{EE}
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight V_{CC} (5V)	Led+
16	Backlight Ground (0V)	Led-

DIMENSIONS in millimeters



LED - H/L B/L	HIGH	LOW
H1	13.2	12.1
H2	8.6	7.5

LCD DESIGN

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), Animations and so on.
A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.
The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. Click to learn more about internal structure of an LCD

6.2.1 DESCRIPTION:

Winstar 16x2 Character LCD Display WH1602W is having two pinout interfaces on upper and bottom sides of the LCD module. This 16x2 LCD display has the outline size of 80.0 x 36.0 mm and VA size of 66.0 x 16.0 mm and the maximum thickness is 13.2 mm. WH1602W 16x2 LCD Displays are built-in controller ST7066 or equivalent. It is optional for + 5.0 V or + 3.0 V power supply. The LEDs can be driven by pin 1, pin 2, or pin 15 pin 16 or A/K. This type of module can be operating at temperatures from -20℃ to +70℃; its storage temperatures range from -30℃ to +80℃.
There are different interface options for WH1602W series, details as below:
►WH1602W: 6800 interface (ST7066 IC)
►WH1602W1: 6800 interface (ST7066 IC)
►WH1602W2: SPI interface (RW1063 IC)
►WH1602W2: I2C interface (RW1063 IC)

6.3. 4-bit and 8-bit Mode of LCD:

The LCD can work in two different modes, namely the 4-bit mode and the 8-bit mode. In 4-bit mode we send the data nibble by nibble, first upper nibble and then lower nibble. For those of you who don't

know what a nibble is: a nibble is a group of four bits, so the lower four bits (D0-D3) of a byte form the lower nibble while the upper four bits (D4-D7) of a byte form the higher nibble. This enables us to send 8-bit data.

Whereas **in** 8-bit mode we can send the 8-bit data directly in one stroke since we use all the 8 data lines.

Now you must have guessed it, yes 8-bit mode is faster and flawless than 4-bit mode. But the major drawback is that it needs 8 data lines connected to the microcontroller. This will make us run out of I/O pins on our MCU, so 4-bit mode is widely used. No control pins are used to set these modes. It's just the way of programming that change.

6.4 Read and Write Mode of LCD:

As said, the LCD itself consists of an Interface IC. The MCU can either read or write to this interface IC. Most of the times we will be just writing to the IC, since reading will make it more complex and such scenarios are very rare. Information like position of cursor, status completion interrupts etc. can be read if required, but it is out of the scope of this tutorial.

The Interface IC present in most of the LCD is HD44780U**,** in order to program our LCD, we should learn the complete datasheet of the IC.

6.4.1 LCD Commands:

There are some preset commands instructions in LCD, which we need to send to LCD through some microcontroller. Some important command instructions are given below:

6.4.2 Command to LCD Instruction Register

0F-LCD ON, cursor ON

01-Clear display screen

02-Return home

04-Decrement cursor (shift cursor to left)

06-Increment cursor (shift cursor to right)

05-Shift display right

07-Shift display left

0E-Display ON, cursor blinking

80-Force cursor to beginning of first line

C0-Force cursor to beginning of second line

38-2 lines and 5×7 matrix

83-Cursor line 1 position 3

3C-Activate second line

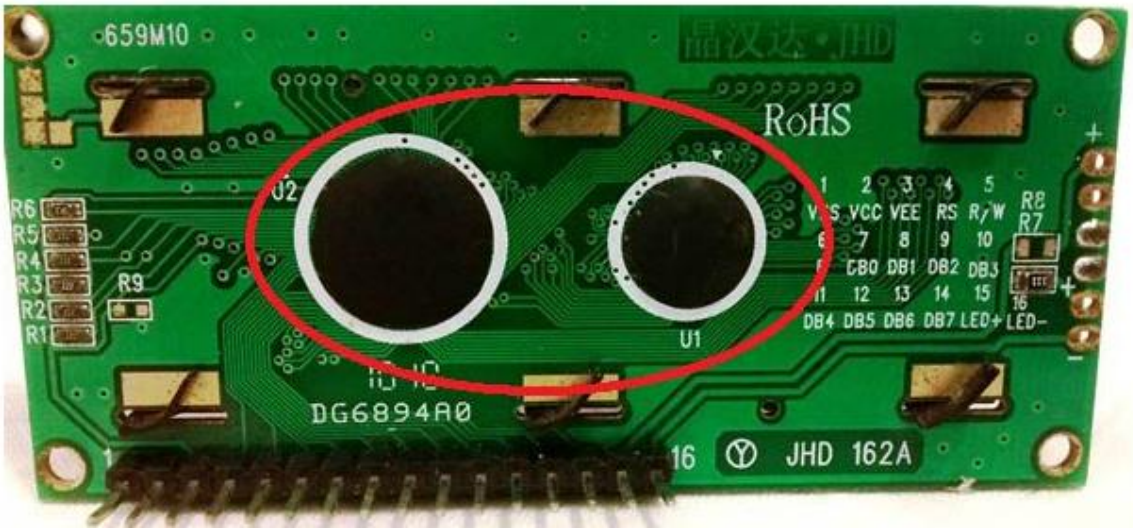
08-Display OFF, cursor OFF

C1-Jump to second line, position 1

OC-Display ON, cursor OFF

C1-Jump to second line, position 1

C2-Jump to second line, position 2

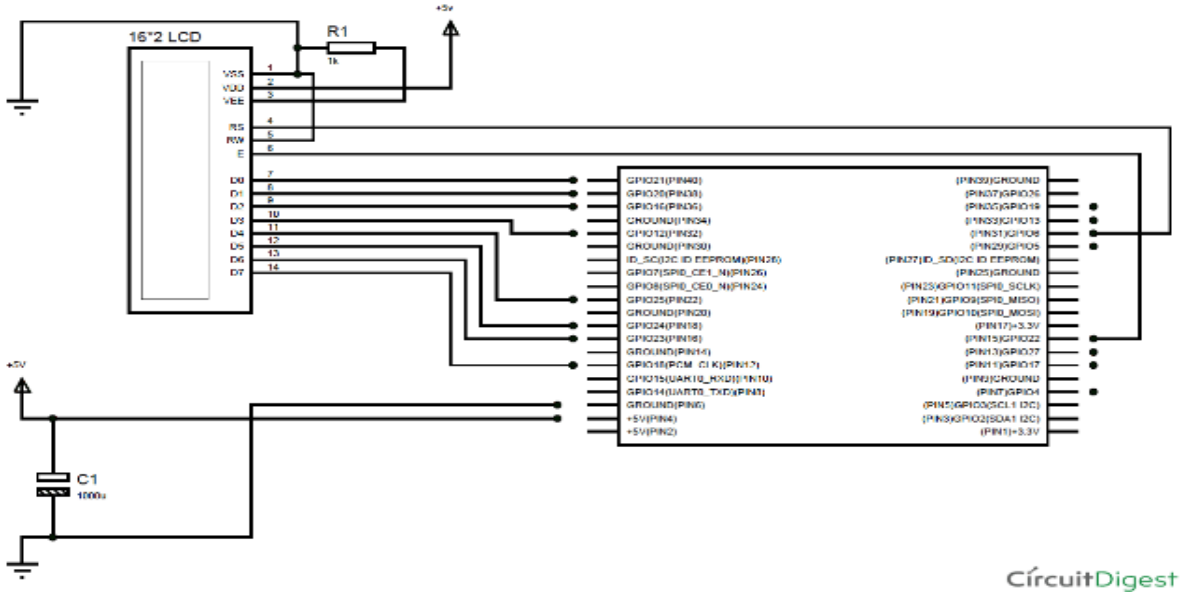


These black circles consist of an interface IC and its associated components to help us use this LCD with the MCU. Because our LCD is a 16*2 Dot matrix LCD and so it will have (16*2=32) 32 characters in total and each character will be made of 5*8 Pixel Dots.

Each character has (5*8=40) 40 Pixels and for 32 Characters we will have (32*40) 1280 Pixels. Further, the LCD should also be instructed about the Position of the Pixels.

It will be a hectic task to handle everything with the help of MCU, hence an **Interface IC like HD44780** is used, which is mounted on LCD Module itself. The function of this IC is to get the **Commands and Data** from the MCU and process them to display meaningful information onto our LCD Screen.

6.5 SCHEMATIC REPRESENTATION:



As shown in the Circuit Diagram, we have Interfaced Raspberry Pi with LCD display **by** connecting 10 GPIO pins of PI to the 16*2 LCD's Control and Data Transfer Pins. We have used GPIO Pin 21, 20, 16, 12, 25, 24, 23, and 18 as a BYTE and created 'PORT' function to send data to LCD. Here GPIO 21 is LSB (Least Significant Bit) and GPIO18 is MSB (Most Significant Bit).

6.6 Process of sending data to LCD:

1. E is set high (enabling the module) and RS is set low (telling LCD we are giving command)
2. Giving value 0x01 to data port as a command to clear screen.
3. E is set high (enabling the module) and RS is set high (telling LCD we are giving data)
4. Proving the ASCII code for characters need to be displayed.
5. E is set low (telling LCD that we are done sending data)
6. Once this E pin goes low, the LCD process the received data and shows the corresponding result. So, this pin is set to high before sending data and pulled down to ground after sending data.

6.7 Specifications:

- **Resolution** The resolution of an LCD is expressed by the number of columns and rows of pixels (e.g., 1024×768). Each pixel is usually composed 3 sub-pixels, a red, a green, and a blue one. This had been one of the few features of LCD performance that remained uniform among different designs. However, there are newer designs that share sub-pixels among pixels and add Quattron which attempt to efficiently increase the perceived resolution of a display without increasing the actual resolution, to mixed results.
- **Spatial performance:** For a computer monitor or some other display that is being viewed from a very close distance, resolution is often expressed in terms of dot pitch or pixels per inch, which is consistent with the printing industry. Display density varies per application, with televisions generally having a low density for long-distance viewing and portable devices having a high density for close-range detail. The Viewing Angle of an LCD may be important depending on the display and its usage, the limitations of certain display technologies mean the display only displays accurately at certain angles.
- **Temporal performance:** the temporal resolution of an LCD is how well it can display changing images, or the accuracy and the number of times per second the display draws the data it is being given. LCD pixels do not flash on/off between frames, so LCD monitors exhibit no refresh-induced flicker no matter how low the refresh rate. But a lower refresh rate can mean visual artefacts like ghosting or smearing, especially with fast moving images. Individual pixel response time is also important, as all displays have some inherent latency in displaying an image which can be large enough to create visual artifacts if the displayed image changes rapidly.
- **Color performance**: There are multiple terms to describe different aspects of color performance of a display. Color gamut is the range of colors that can be displayed, and color depth, which is the fineness with which the color range is divided. Color gamut is a relatively straight forward feature,

but it is rarely discussed in marketing materials except at the professional level. Having a color range that exceeds the content being shown on the screen has no benefits, so displays are only made to perform within or below the range of a certain specification. There are additional aspects to LCD color and color management, such as white point and gamma correction, which describe what color white is and how the other colors are displayed relative to white.

- **Brightness and contrast ratio:** Contrast ratio is the ratio of the brightness of a full-on pixel to a full-off pixel. The LCD itself is only a light valve and does not generate light; the light comes from a backlight that is either fluorescent or a set of LEDs. Brightness is usually stated as the maximum light output of the LCD, which can vary greatly based on the transparency of the LCD and the brightness of the backlight. In general, brighter is better, but there is always a trade-off between brightness and power consumption

6.9 Applications:

- Calculators
- Watches
- Clocks
- Telephones
- Computer Monitors (Computer Screens)
- Instrument Panels
- Television

Chapter-7

Raspberry PI with Raspbian OS

Introduction:

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

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

- Below are some of the Operating systems that a Pi can run:

Linux	There are three official Linux flavours available for download namely
	Debian [Raspbian] *Recommended
	Arch Linux
	Pidora [Based on Fedora]
RISC OS	A retro looking 1080p GUI designed by the ARM designers. RISC was more common during the 90's
Firefox OS	A new OS by the Firefox team. Pretty much a combination of Firefox and PTXdist-built Linux
Plan 9	Unix like OS by the by the Bell Labs, created by the UNIX creators
Android	No explanation necessary, but this hasn't gone beyond a 2.3 build and a bit too slow.

- Raspbian OS is one of the official Operating systems available for free to download and use. The system is based on Debian Linux and is optimized to work efficiently with the Raspberry Pi computer. As we already know an OS is a set of basic programs and utilities that runs on a specified hardware, in this case the Pi. Debian is very lightweight and makes a great choice for the Pi. The Raspbian includes tools for browsing, python programming and a GUI desktop.
- The Raspbian desktop environment is known as the “Lightweight X11 Desktop Environment” or in short LXDE. This has a fairly attractive user interface that is built using the X Window System software and is a familiar point and click interface.

- Raspbian uses PIXEL, Pi Improved Xwindows Environment, Lightweight as its main desktop environment as of the latest update.
- It is composed of a modified LXDE desktop environment and the Open box stacking window manager with a new theme and few other changes.
- The distribution is shipped with a copy of computer algebra program Mathematica and a version of Minecraft called Minecraft Pi as well as a lightweight version of Chromium as of the latest version.

7.1 Basic features:

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

7.2 Setting Up Raspbian OS:

Let's first connect the board with all the necessary accessories to install and run an operating system.

Step 1: Take the Pi out of its anti-static cover and place it on the non-metal table.

Step 2: Connect the display – Connect the HDMI cable to the HDMI port on the Pi and the other end of the HDMI cable to the HDMI port of the TV.

Step 3: Connect your Ethernet cable from the Router to the Ethernet port on the Pi

Step 4: Connect your USB mouse to one of the USB ports on the Pi

Step 5: Connect your USB Keyboard to the other USB port on the Pi

Step 6: Connect the micro USB charger to the Pi but don't connect it to the power supply yet

Step 7: Flash the SD Card with the Raspbian OS.

1. To prepare the car for use with the Pi we will need to put a OS on the card. We certainly cannot drag and drop the OS files on to the card but the flashing the card is not too difficult either.

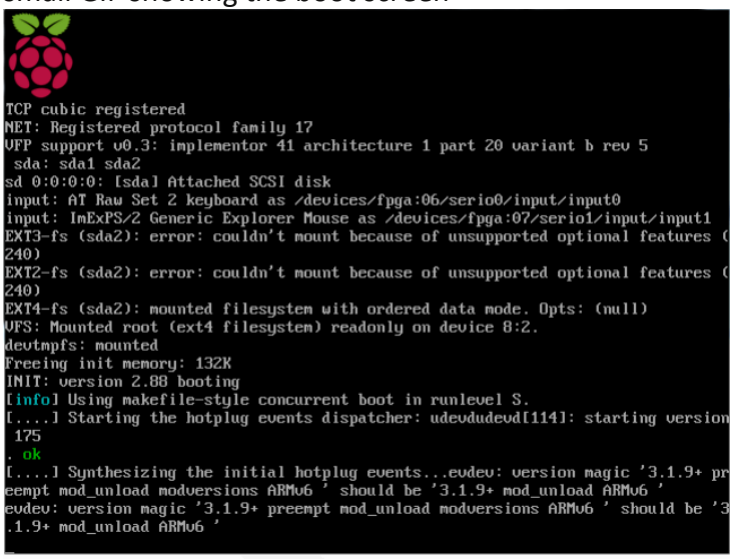
2. Since we have already decided to install Raspbian, lets download the RASPBIAN image from the following link. http://www.raspberrypi.org/downloads/.
3. Unzip the contents of the Zip file into a folder on your machine, one of the unzipped files would be a .img file which is what needs to be flashed on to the SD card. [In case there are more than one file, the current version of the zip has only this file and none other]
4. Flashing from Linux instructions.
    1. Start the terminal on your Linux OS
    2. Insert the empty SD Card into the card reader of your machine.
    3. Type sudofdisk -l to see all the disks listed. Find the SD card by its size, and note the device address (/dev/sdX, where X is a letter identifying the storage device. Some systems with integrated SD card readers may use /dev/mmcblkX— format, just change the target in the following instructions accordingly).
    4. Use cd to change to the directory with the .img file you extracted from the Zip archive
5. Type sudo dd if=imagefilename.img of=/dev/sdX bs=2M to write the file imagefilename.img to the SD card connected to the device address. Replace imagefilename.img with the actual name of the file extracted from the Zip archive. This step takes a while, so be patient! During flashing, nothing will be shown on the screen until the process is fully complete.

1.
```
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x24282427

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1       12748   102398278+   7  HPFS/NTFS
/dev/sda2           12749       25496   102398310    5  Extended
/dev/sda3           25497       77825   420332692+   7  HPFS/NTFS
/dev/sda5           12749       12997     2000061   82  Linux swap / Solaris
/dev/sda6           12998       25496   100398186   83  Linux

Disk /dev/sdb: 3965 MB, 3965190144 bytes
49 heads, 48 sectors/track, 3292 cylinders
Units = cylinders of 2352 * 512 = 1204224 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               4        3293     3868160    b  W95 FAT32
blacklaw@xerxes-linux:/media/Data/raspberrypi/debian6-19-04-2012$ sudo dd if=deb
ian6-19-04-2012.img of=/dev/sdb bs=2M
```

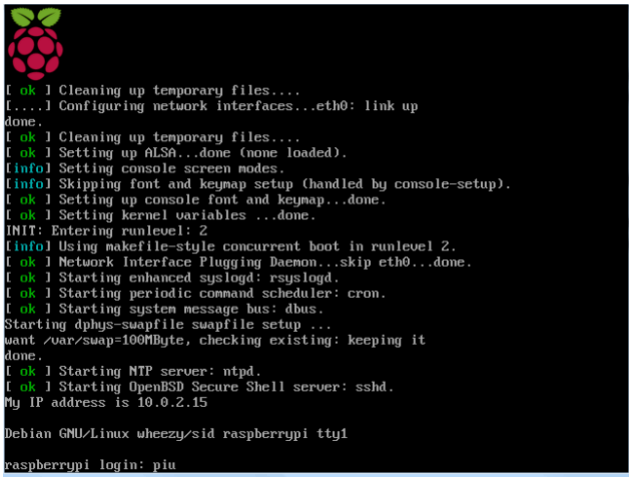
6. Flashing from Windows Instructions:
    1. The Image Writer for Windows is used in place of dd which designed specifically for creating USB or SD card images of Linux distributions, it features a simple graphical user interface that makes the creation of a Raspberry Pi SD card straight forward. Download the latest version of Image Writer for Windows from the website: **https://launchpad.net/win32-image-writer.** Below are the steps.

i. Download the binary (not source) Image Writer for Windows Zip file, and extract it to a folder on your computer.
ii. Plug your blank SD card into a card reader connected to the PC.
iii. Double-click the Win32DiskImager.exe file to open the program, and click the blue folder icon to open a file browse dialogue box.
iv. Browse to the imagefilename.img file you extracted from the distribution archive, replacing imagefilename.img with the actual name of the file extracted from the Zip archive, and then click the Open button.
v. Select the drive letter corresponding to the SD card from the Device drop-down dialogue box. If you're unsure which drive letter to choose, open My Computer or Windows Explorer to check.
vi. Click the Write button to flash the image file to the SD card.

2. Once the OS is flashed, insert the SD card into the Pi SD Card slot
3. Connect the Micro-USB to the power source and switch it on.
4. Now the system boots into the below screen and the LED's on the board will Blink. Below is a small GIF showing the boot screen



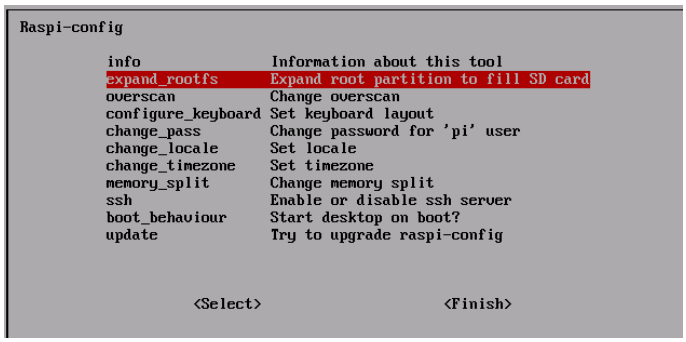
5. Now you will need to login with username/password combination of pi/raspberry.
6. If you would like to use the GUI interface type startx. Below is the image showing the previous two steps.



You are now all set with the OS installed and your Pi up and running.

7.3 Configuring your Raspbian:

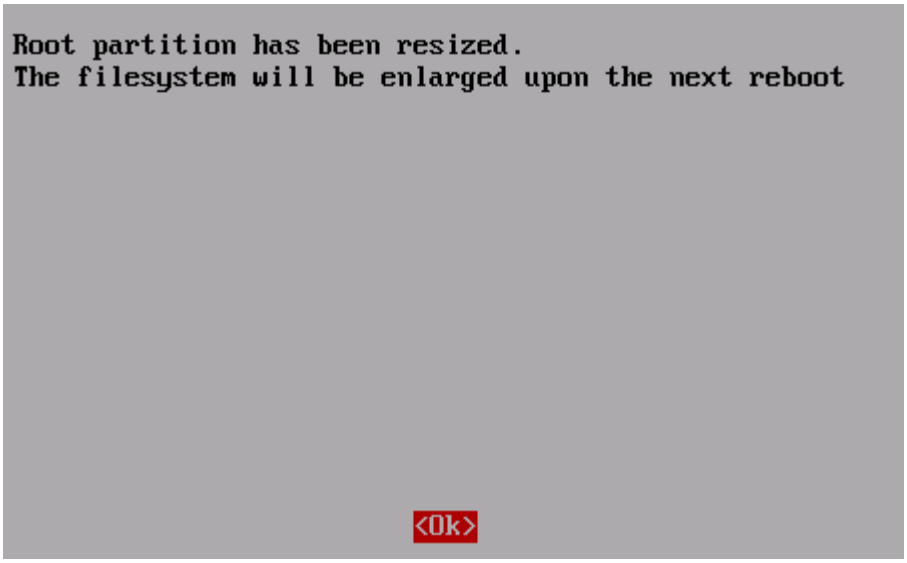
If you need to enter the config menu of the Pi, you will need to use the sudo raspi-config command. This config appears on the first boot of your Pi as well. Below are some options available and their quick explanations.



Some of the booting options are

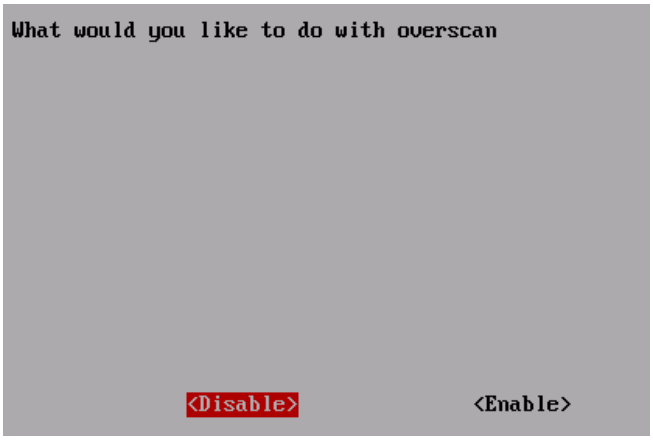
Step 1: expand_rootfs.

Choose or highlight the “expand_rootfs” option and press enter. Use your keyboard arrow keys to select the option. Once you press enter you will get a confirmation screen like below, press enter again and you will be taken back to the main config screen.



Step2: over scan:

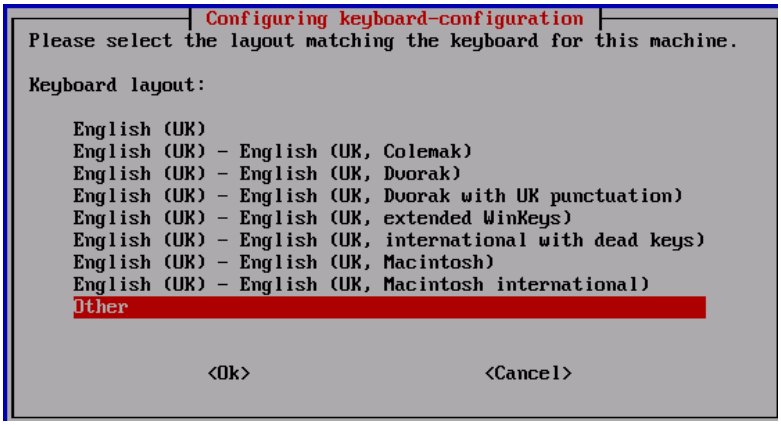
You can either enable or disable over scan here. I am disabling it so my screen fills up completely.

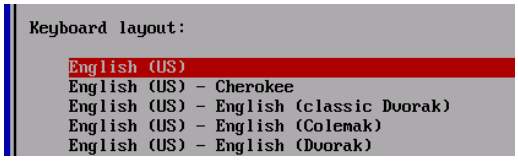


Step 3: configure keyboard:

Since the initial setting gives me the UK keyboard format, I will change the setting to use the generic international 105 with English (US) option. Select "configure keyboard" and press Enter.

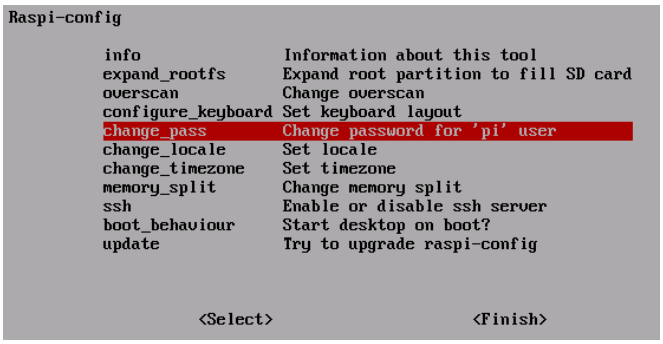
Once the keyboard type is set, you'll need to specify the layout. There's a good chance you want a different layout than English (UK) like me, so choose "Other" and select an appropriate option for you. I am sticking with English (US).





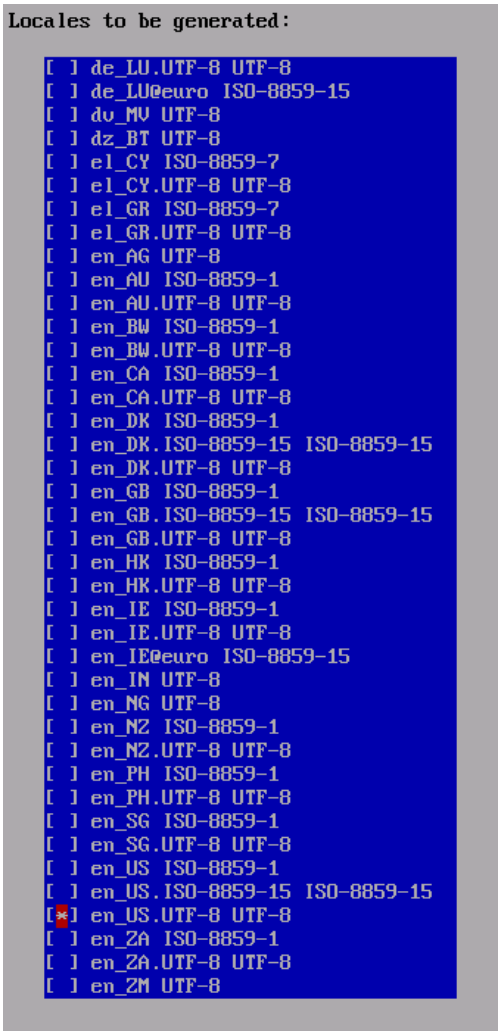
After this just choose the default modifier keys when asked, as well as "No compose key" on the next screen. If later you find you need a compose key to create alternative characters, you can return to this configuration screen by running "raspi-config".

Step 4: Change password. Select change pass and the rest are self-explanatory steps. I think it’s a good idea to change the generic password :)



Step 5: Setting up Locale:

If you live in the USA then you want en_US.UTF-8 else it depends on where you live. Scroll down to your locale you need, and de-select the en GB [Great Britain] option on your way. In our case, we'll be enabling en_US.UTF-8

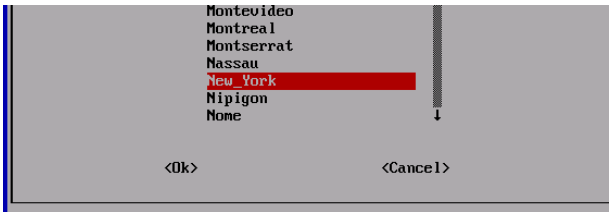


The next dialogue window will ask you to choose a default locale, select the locale you just chose on the previous screen and press Enter.

Step 6: Setup Time zone:

Select the "change_timezone" option. You'll be presented with a list of regions first.

The next dialogue will show you a list of zones within that region. I am sure you can handle it from here J



Step 7: Finish

Go back to the main menu and select Finish. This should take you back to the original boot screen and you can type startx from there to enter the LXDE environment or the GUI.

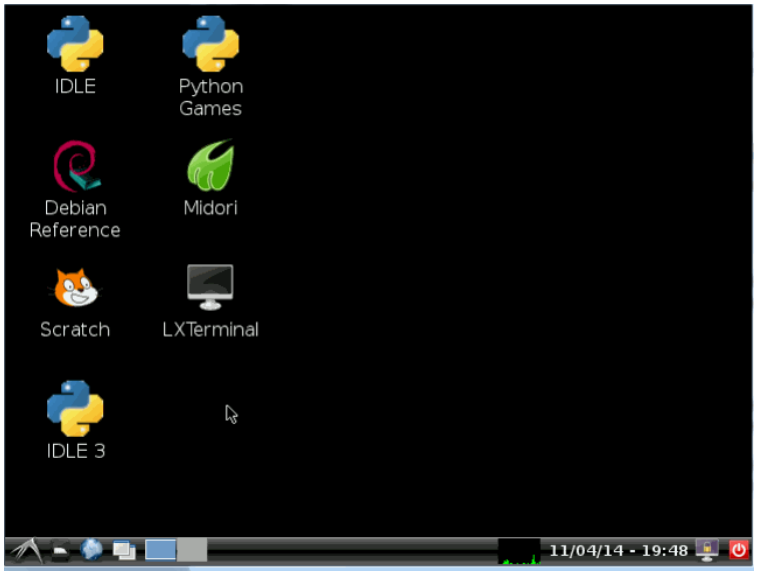
7.3 File system layout:

As most of you already know a File System is the chosen data structure and access methods used by any operating system to store, organize and access files. File System is also used to denote the partitions on the disk that is being used on a machine. Every filesystem has its own method of storing files onto a storage medium [like a hard disk or SD card]. Without these representations which are recognized it would be hard to share file between systems and people.

7.3.1 Logical layout:

The Linux way of storing and organizing files is a bit different than that of Windows. Unlike Windows where everything is stored under drives and each drive having a letter, in Linux everything is grouped under a branch under the root file system.

To take a look at what these branches are, open the Terminal on your pi and type the below command



As you can see in the above gif image, there are various directories. Some of those directories on the SD Card for persisting files and others are the virtual directories for accessing different portions of the operating system or the hardware. Below are the list of what you see and a short description.

Directory	Description
Boot	Folder for Linux Kernel and other packages necessary to booth/start the Pi
Bin	The Raspbian related binary files including those for required to run the GUI of the OS
Dev	This is one of those virtual directories and this is used for accessing all the connected devices including the storage
Etc	Misc. config files like encrypted passwords are put in this
Home	This is like My documents and each username will get a separate directory under this
Lib	Lib= libraries, code required by various applications
lost+found	Dump of pieces of files are stored here when the system crashes
Media	Dir for removable storage dives like USB and CD
Mnt	Used to mount external hard drives and similar storage devices manually
Opt	Optional software directory, any apps that are not part of the OS will go here
Proc	Another Virtual directory, this contains info about running processes (programs)
Selinux	Security utilities originally developed by our beloved NSA and these are related to Security Enhanced Linux
Sbin	System maintenance binaries typically used by the root/superuser
Sys	Operating system files
Tmp	Temporary files
Usr	This is used as storage for user accessible programs
Var	Virtual directory that a program can use to persist data

7.3.2 Physical Layout:

Even though we see all those directories above, if you look at the SD card contents you will see a completely different structure.

For the Raspbian the card will be organized into two sections or partitions.

The first partition is a smaller with around 75 MB and is formatted as VFAT, this is the same format used by Microsoft for the removable drives. This partition is mounted and is accessible by Linux in the /boot directory and this is the folder in which all the files used to configure the Pi and run the OS are stored.

The second partition a larger one and formatted as EXT4, EXT4 is one of the native Linux Filesystems and is considered for its data safety and high speed. This contains the main chunk of the distribution and the programs and user files.

7.4 Software Management on PI:

- sudo apt-get update
- sudo apt-get upgrade
- sudo apt-get install<package>
- sudo apt-get download<package>
- sudo apt-get remove<package>
- sudo apt-get purge<package>
- sudo apt-get dist-upgrade

Chapter-8

THINGSPEAK AND IoT

Introduction:

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize and analyse live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by your devices to ThingSpeak. With the ability to execute MATLAB code in ThingSpeak you can perform online analysis and processing of the data as it comes in. ThingSpeak is often used for prototyping and proof of concept IoT systems that require analytics.

8.1 IoT:

Internet of Things (IoT) describes an emerging trend where a large number of embedded devices (things) are connected to the Internet. These connected devices communicate with people and other things and often provide sensor data to cloud storage and cloud computing resources where the data is processed and analysed to gain important insights. Cheap cloud computing power and increased device connectivity is enabling this trend.

The term “Internet of Things” (IoT), coined by Kevin Ashton in 1999 [2], has been in use for several years and continues to be of interest, specifically when it comes to technological progress.

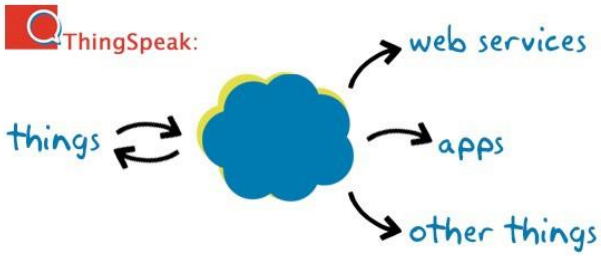
The term “Internet of Things” (IoT), coined by Kevin Ashton in 1999 [2], has been in use for several years and continues to be of interest, specifically when it comes to technological progress

8.2 Operating Principals of IoT And Usage Of thingsspeak:

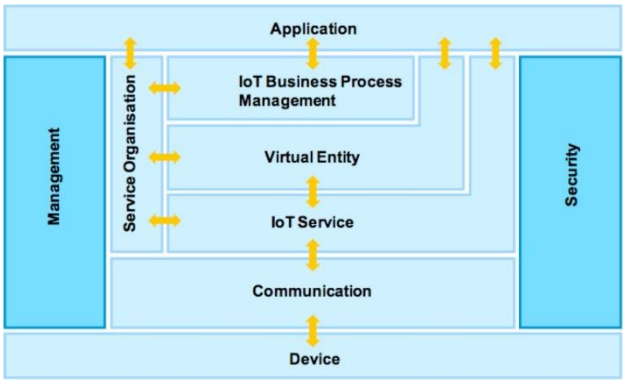
In order to connect an object to the IoT, several things are needed in the hardware and software realm. First of all, if one wishes to go beyond simply connecting data from a computer, objects to gather (sensors) or receive (actuators) data are necessary. For example, a digital thermometer can be used to measure temperature. In this case, the data needs to be uploaded to a network of connected servers which run applications. Such a network is commonly referred to as ‘the cloud’.

In case of IoT, an object will connect to the cloud through a (possibly wireless) internet connection to upload or receive data. Objects to be connected are typically augmented with either sensors or actuators. For the purpose of connecting an object to the IoT, we focus on the ThingSpeakAPI. The interface provides simple communication capabilities to objects within the IoT environment, as well as interesting additional applications. Moreover, ThingSpeak allows you to build applications around data collected by sensors. It offers near real-time data collection, data processing, and also simple visualizations for its users. Data is stored in so-called channels, which provides the user with a list of features. Each channel allows you to store up to 8 fields of data, using up to 255 alphanumeric characters each. All incoming data is time and date stamped and receives a sequential ID. Once a channel has been created, data can be published by accessing the ThingSpeakAPI with a ‘write key’, a randomly created unique alphanumeric string used for authentication. Consequently, a ‘read key’ is

used to access channel data in case it is set to keep its data private (the default setting). Channels can also be made public in which case no read key is required.



According to the *ThingSpeak* website, the API works as noted in Figure 1. Essentially, 'things' are objects that are given sensors to collect data. Data is sent and received via simple "Hypertext Transfer Protocol" (HTTP) POSTs, much like going to a web page and filling out a form. This communication happens through plaintext, *JSON* or *XML*. The data is then uploaded to the cloud and from there can be used for a variety of purposes. In turn, data (such as commands or choosing certain options) can be gathered and communicated to the cloud, which in turn sends these messages to the object.



A deeper level of what occurs, especially on the server side. When a device sends data through a HTTP request (communication), it is processed by the IoT service (in this case *ThingSpeak*), which communicates with a virtual server. Both the server and the IoT service communicate directly with the application. Finally, at all levels of communication from the device to the application there is both requirements regarding security and management of the data transfer. Unfortunately, *ThingSpeak* does not document how the specific parts of the diagram are handled on a technical level.

8.3 ThingSpeak Strengths

The key thing which separates *ThingSpeak* from any of the mentioned competitors is that it creates a sense of community through the possibility of creating public channels. It is noted to be the only open data platform specifically designed for the IoT in 'the cloud'. Also, the API allows for very easy visualization of collected data through using spline charts. Therefore, it is visually appealing and is much easier when examining collected data compared to other open source APIs.

Another point in ThingSpeak's favor is the fact that it uses Phusion Passenger Enterprise, a web and

application server. Therefore, the API provides additional support for the programming languages Ruby, Python andNode.js.

These languages are noted to be powerful and popular, so additional support and features for these are seen as a nice supplement. Although the other open source APIs are able to work with most languages, extra features for the most popular ones are always a bonus.

Unlike *Carriots*, the APIs *ThingSpeak*, *SmartObject* and *SensorThings*are open source and are therefore able to be integrated with any hardware device, including *Arduino*, *Raspberry Pi* and any home-made micro-controller. In addition, anyone is able to develop the platform, so if a desired feature does not exist, anybody is free to code it. Therefore, open-source APIs that allow connectivity with any micro-controller would be the ideal ones to choose between when conducting any "do-it-yourself" (DIY) project.

Another benefit of having an open source API is that it can be run locally or on your own server. Therefore, a user is provided with a lot of flexibility and control for their projects. The advantage of *ThingSpeak* over the others however, is that free hosting is provided for data channels. As mentioned earlier, these channels can be private or public. Not only does this make it easier for someone who does not operate and maintain a server, but the use of HTTP POST and GET for the data is relatively easy for newcomers to web technology. Lastly, public channels also serve as a source of inspiration, as users are able to examine and admire the projects of others. ThingSpeak excels at being one of the simpler APIs to get started quickly, as well as the ability to use a community server to host a channel to upload data. It also provides additional features for Ruby, Node.js and Python.

8.4 ThingSpeak Weaknesses

After considering all the strengths, there are some aspects of the ThingSpeakAPI that can be called its weaknesses. For example, when uploading data to the API there is a limit up one update per channel every fifteen seconds.

We postulate that the reason for this limit in uploads is due to the excess bandwidth that could be used, and therefore would end up costing ThingSpeak additional funds for a non-profit service. This limit can be removed by taking the whole API to another web host provider to run the API. Although it is not a weakness in comparison with the competing APIs introduced earlier, it would be nice to have other types of charts available for graphing data, such as pie charts, bar charts and histograms.

However, it is very simple to export the data to other programs that are able to build more complex and colorful charts to display data. As with many open source projects, using the ThingSpeakAPI can be a hurdle for beginners. In other words, ThingSpeak is not necessarily a 'turn key' API, although we have noted it to be slightly simpler than some of its competitors. Consequently, to use ThingSpeak some coding knowledge is needed.

One of the biggest downfalls of the API and using Twitter is the fact that Twitter messages have to be unique, or more specifically two identical messages cannot be used after each other due to spam protection on the Twitter side. When using this extension of ThingSpeak it is necessary to add unique data (such as a timecode) to ensure undisrupted operation. It is perhaps not necessarily a limitation of

ThingSpeak itself but something that should be kept in mind when working with one of its features.

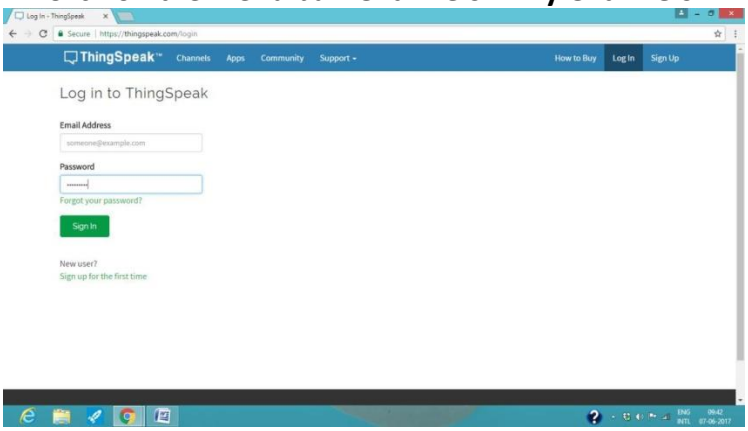
As a Last note, we have to mention something about privacy. Channels are by default set on private and users or machines need a read or write key to access the data. Changing the channel to public makes the data available for everyone without the need for a read key. ThingSpeak does not inform on their website where the data is stored and how it is secured. It is also not known for what duration data is stored on their servers

8.5 Procedure For creating Channels:

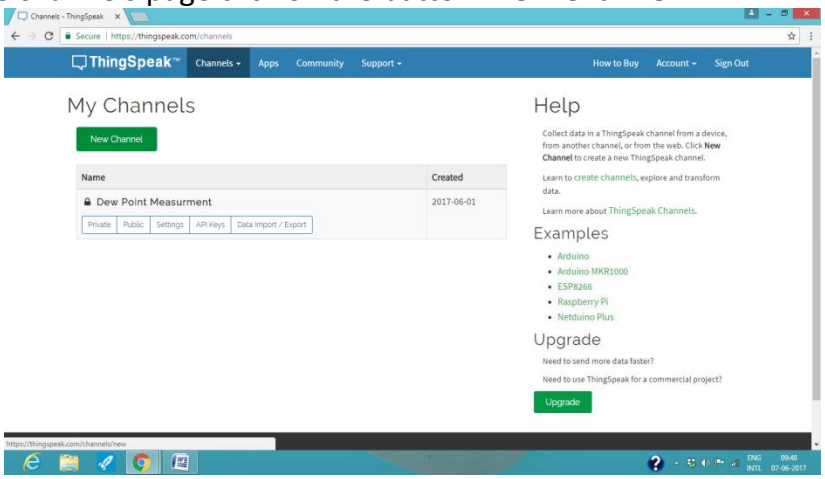
Before creating a channel, you need to sign in to things speak. You can easily **sign in** either using your either **thingspeak account** or **mathworks account**, or **create a new mathworks account** via following link:

https://thingspeak.com/users/sign_up

1. Click on the menu bar '**Channels > My Channels**.

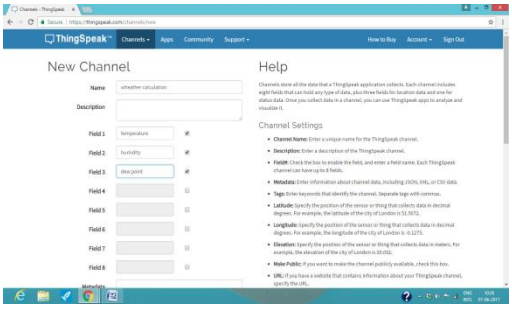


2. Now on the channels page click on the button '**New Channel'**.



3. Newchannelpagehavevarioustextboxfieldsshowingthesettingsofthechannel.
    a. **Name** – provide a unique name to your channel.

b. **Fields** – Click the check boxes next to the field and then enter the fieldname.

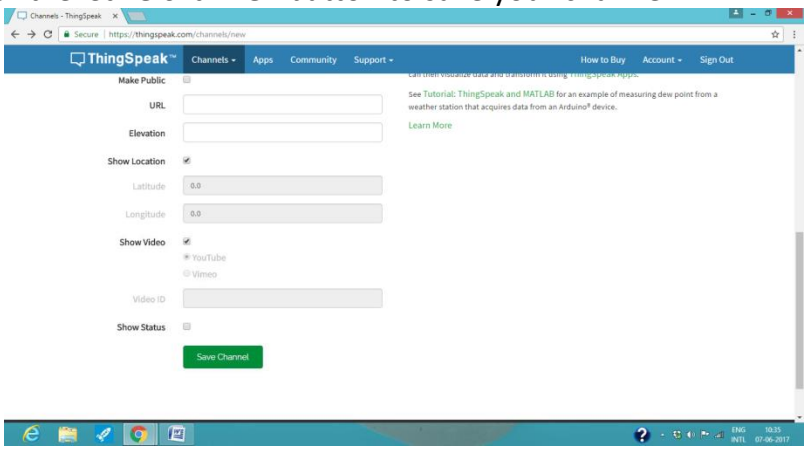


c. To make your channel public check the '**Make Public** 'checkbox.

d. Similarly, you can also add the location to your channel by clicking the '**Show Location'** checkbox.

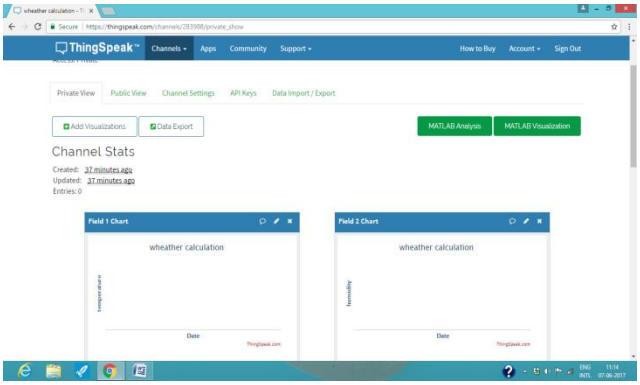
e. Check the '**Show video**' check box to make the video visible uploaded by you.

f. Now click the '**Save channel**' button to save your channel.

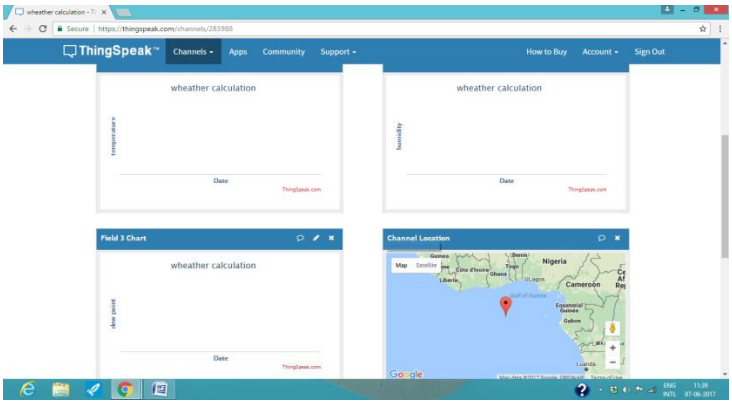


4. Now, the channel page opens with the following tabs:

a. **Private View**- It displays the information about your channel that is only visible to you.

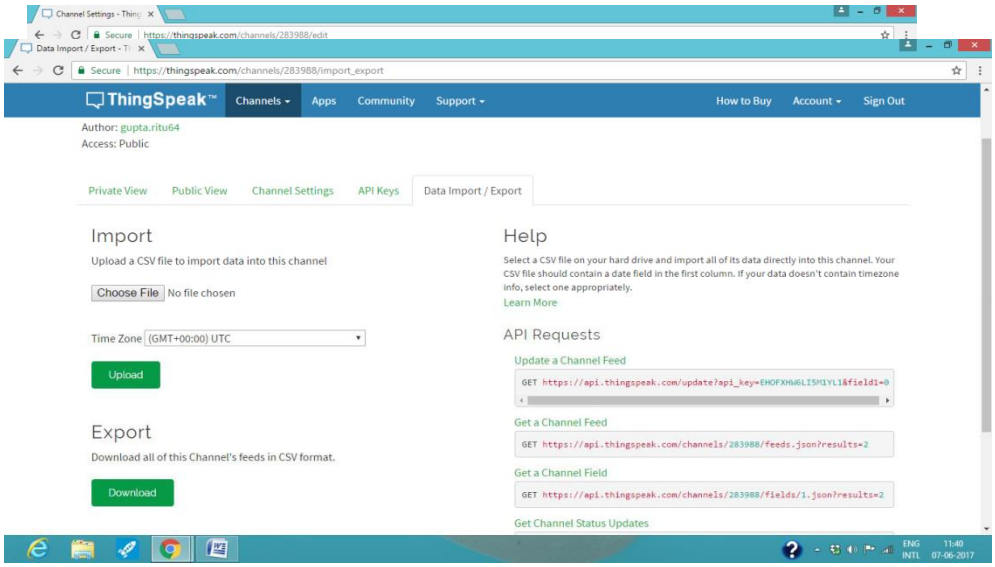


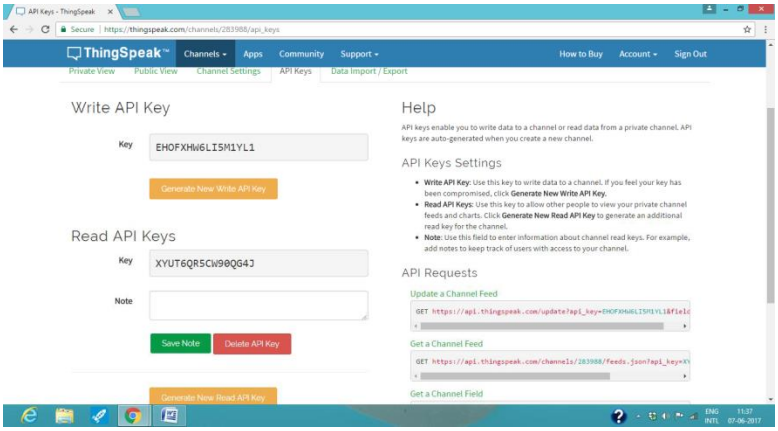
b. **Public View**- If you have chosen to make your channel publicly visible then it will display the selected fields and information



c. **Channel Settings**- It will show all the options that are available during the channel creation.

d. **API Keys**- In this tab you will have two API Keys – Read API key (to read from your channel), write API Key (to write to your channel).





e. **Data Import/export**- It enables you to import and export the channel data.

5. In future your channel will be available to you just by clicking **'Channels>My Channels'**.

8.6 TYPICAL APPLICATIONS

Although the original intended use of the *ThingSpeak*API is to 'give voice' to everyday objects, it seems that there is a very common emerging trend on what the users are building and sharing with the IoT. Scholars note that the IoT will be most useful in an organizational environment, especially for inventory management, production efficiency, waste management, urban planning, environmental sensing, social interaction gadgets, continuous care, emergency response, smart product management, as well as other uses focusing on creating an efficient and sustainable urban environment. For individuals and private homes, the IoT is predicted to incorporate smart metering of electricity, home automation and intelligent shopping. Overall then, the typical applications of any API used to connect objects to the IoT is broad with far-reaching implications. More specifically though, *ThingSpeak* is an open source API. Therefore, we expect much more 'home/private' use applications rather than organizational as this is the typical way that new, open source technologies are first put to use. Without a doubt, the majority of public channels are focused on sensors in homes. Interestingly, most of these were measuring weather data, especially temperature (indoor/outdoor), humidity, light levels and atmospheric pressure. Hence, most *ThingSpeak* users choose to apply the API for personal measures. That said, it was not completely uncommon for *ThingSpeak* to be applied to organizations and some of those used it to monitor the temperature and humidity of an office.

A second common use found for *ThingSpeak* is still connected to light measurement, but not to determine how bright a room is. Instead, ThingSpeak was used to measure the amount of energy created by a photovoltaic panel. This particular panel was located in a remote field in Germany. It is not uncommon for roof space to be rented in rural areas for placement of such panels where one can be paid for feeding the public power grid. Therefore, many owners of photovoltaic panels are nowhere near their placement, which makes it critical to be able to monitor the power generation from a remote location. The ability to monitor panels allows owners to see which have the most ideal placement, and also if there are any technical errors with the panels.

Furthermore, still in line with measuring the environment,

ThingSpeak has also been used to measure and control the temperature of a water bed. This is interesting since one can note the ideal temperature for getting into bed, and also perhaps be able to detect when the bed is in use by (unwanted) inhabitants, such as pets. Hence, there are many

different uses and necessities for measuring changes in one's environment.

8.7 Thingsspeak with python:

8.7.1 Installation:

1. Pip Install thingspeak API:

To install thingspeak, simply run this simple command in your terminal of choice:

$ pip install thingspeak

2.Get the Source Code

Requests is developed on GitHub, where the code is always available.
You can either clone the public repository:
$ git clone git@github.com:mchwalisz/thingspeak.git
Or, download the tar ball:
$ curl -OL https://github.com/mchwalisz/thingspeak/tarball/master
 # optionally, zipball is also available (for Windows users).
Once you have a copy of the source, you can embed it in your own Python package, or install it into your site-packages
easily:

$ python setup.py install

8.7.2 Basic Usage:

1.Command line

It is possible to view the channel directly:
$ thingspeak -q -r 2 9

{
"channel": {
"created_at": "2010-12-14T01:20:06Z",
"description": "Netduino Plus connected to sensors around the house",
"field1": "Light",
"field2": "Outside Temperature",
"id": 9,
"last_entry_id": 9680334,
"latitude": "40.44",
"longitude": "-79.9965",
"name": "my_house",
"updated_at": "2016-02-09T20:11:45Z"
},

"feeds": [
{
"created_at": "2016-02-09T20:11:31Z",
"entry_id": 9680333,
"field1": "199",
"field2": "29.978768577494691"
},
{
"created_at": "2016-02-09T20:11:45Z",
"entry_id": 9680334,
"field1": "213",
"field2": "29.723991507430998"
}
]
}

2.Python interface

Or through Python interface:

>>>importthingspeak

>>>ch=thingspeak.Channel(9)

>>>ch.get({'results':2})

u'{"channel":{"id":9,"name":"my_house",

"description":"Netduino Plus connected to sensors around the house",

"latitude":"40.44","longitude":"-79.9965",

"field1":"Light","field2":"Outside Temperature",

"created_at":"2010-12-14T01:20:06Z",

"updated_at":"2016-02-09T20:13:45Z","last_entry_id":9680342},

"feeds":[

{"created_at":"2016-02-09T20:13:30Z","entry_id":9680341,

"field1":"199","field2":"29.554140127388536"},

{"created_at":"2016-02-09T20:13:45Z","entry_id":9680342,

"field1":"193","field2":"27.855626326963908"}]}

8.7.3API

class thingspeak.**Channel**(*id*, *api_key=None*, *fmt='json'*, *timeout=None*, *server_url='https://api.thingspeak.com'*)[source]

ThingSpeak channel object

get(*options=None*)[source]

Get a channel feed.

get_field(*field=None*, *options=None*)[source]

Get particular field

get_field_last(*field=None*, options=None*)[source]

To get the age of the most recent entry in a channel's field feed

get_last_data_age(*field=None*, options=None*)[source]

Get last result from particular field in text format

update(*data*)[source]

Update channel feed.

view()[source]

View a Channel

Chapter-9

Results

Introduction:

- Here we are using Python language here for the Program. Before coding, user needs to configure Raspberry Pi.
- Also, here we are using MobaXterm (A Server) to Configure raspberry pi in Laptop.
- Unlike the normal configuration of Pi3 on PC Monitor, it involves a different set of process
- After configuring the laptop with raspberry pi3 we will install all the required packages (includes Sensor libraries) which we will be using while doing coding.

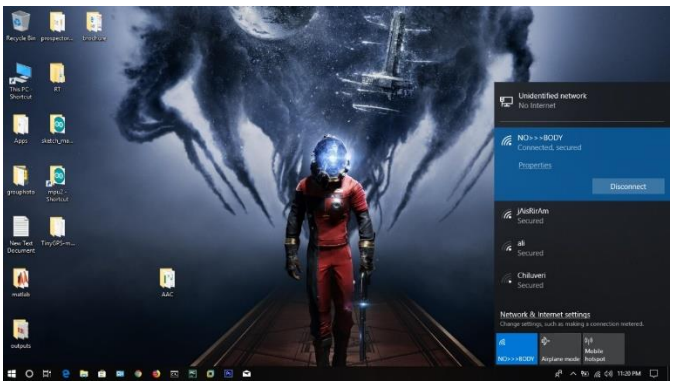
9.1 All About MobaXterm and Installation of basic packages:

MobaXterm is your ultimate toolbox for remote computing. In a single Windows application, it provides loads of functions that are tailored for programmers, webmasters, IT administrators and pretty much all users who need to handle their remote jobs in a simple fashion.

MobaXterm provides all the important remote network tools (SSH, X11, RDP, VNC, FTP, MOSH, ...) and Unix commands (bash, ls, cat, sed, grep, awk, rsync, ...) to Windows desktop, in a single portable exe file which works out of the box.

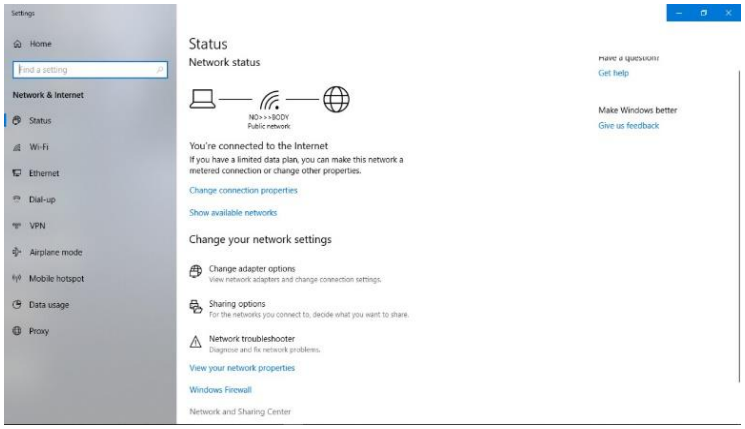
We need to follow the following procedure to configure Pi3 with Laptop:

Step1: Connect to any Network (LAN or Wi-fi) through which data transfer takes place.

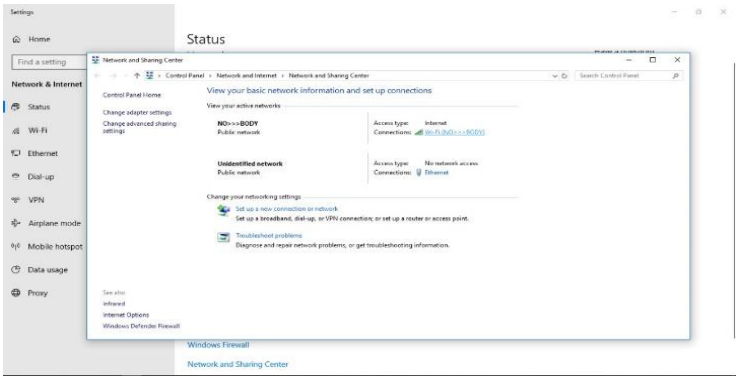


Step2: In this step we share our network with raspberry pi via Ethernet

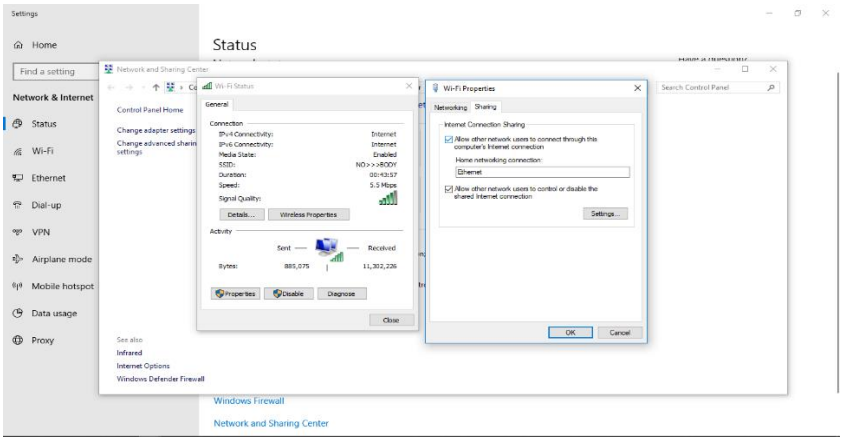
First Open the Network Status, check whether connected to any network or not. Then go to Network & Internet Settings.



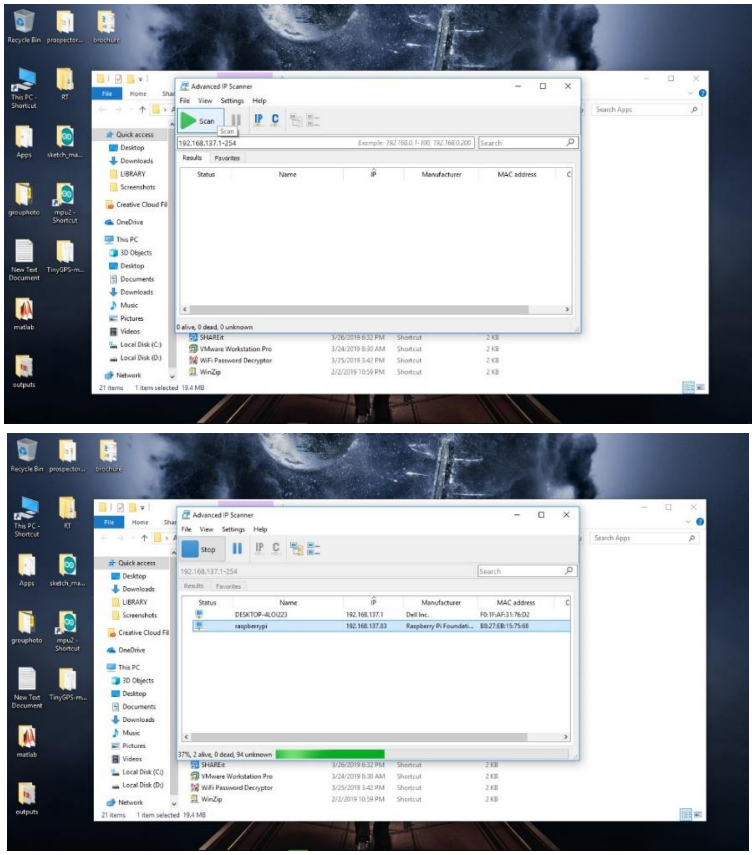
Now, go to Network and Sharing Center as shown below.



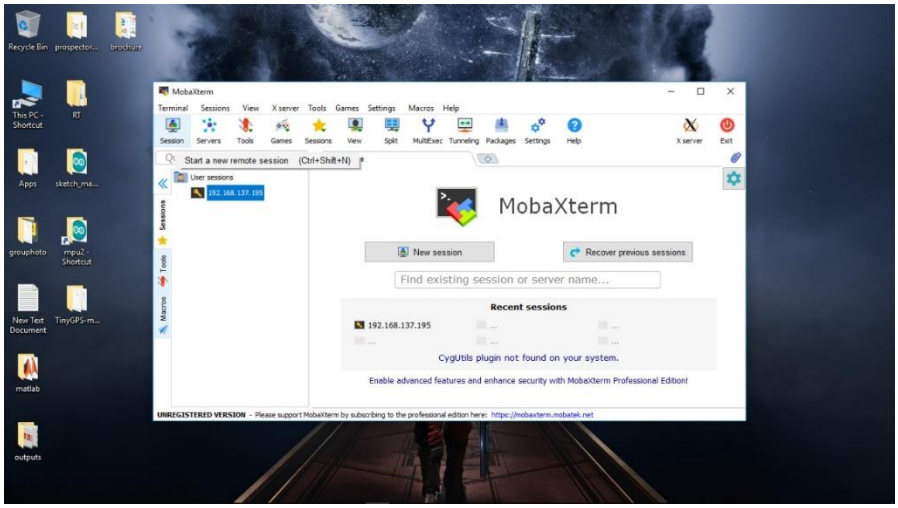
Then, click on the Main network(wi-fi) and a dialogue box get opens as below. In that box, below click on Properties. And in Properties on the top, go to Sharing and check-in as shown below.



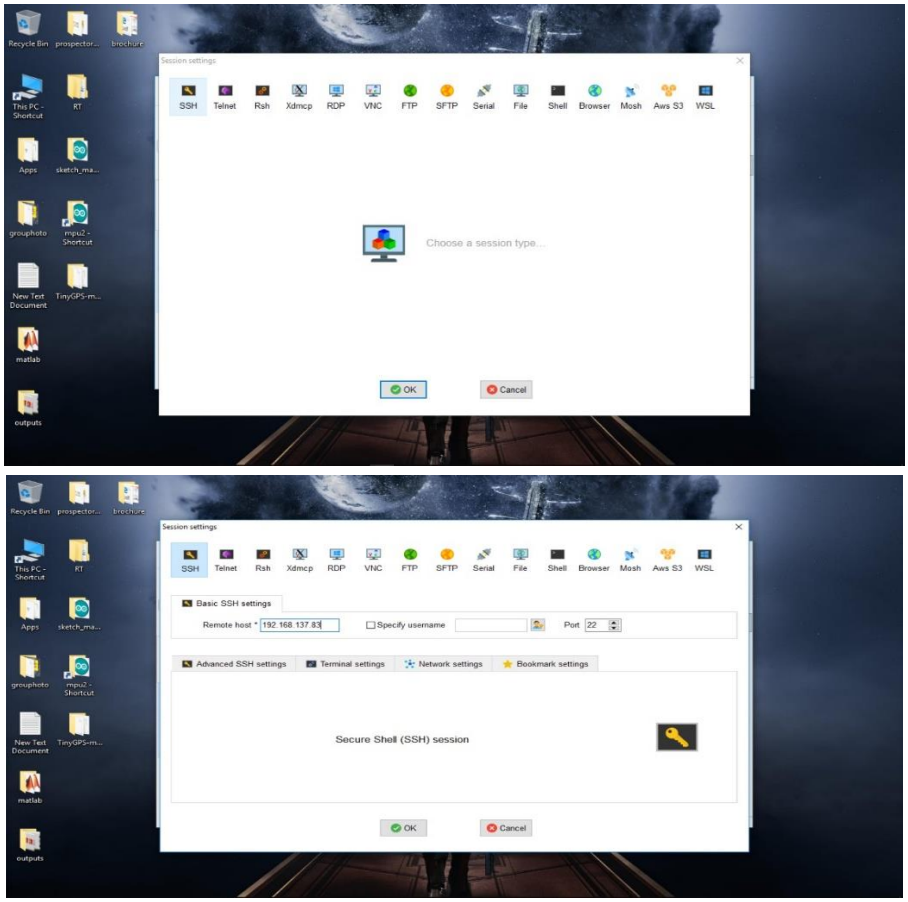
Step3: Now when we have connected the PI3 and desktop to same network, we need to find the IP Address allocated to PI3 in-order to configure it. Here we will be using Advanced IP Scanner to find the IP Address of PI3.



Step4: Now, we will launch MobaXterm for the configuration of Pi3 using it's IP address.

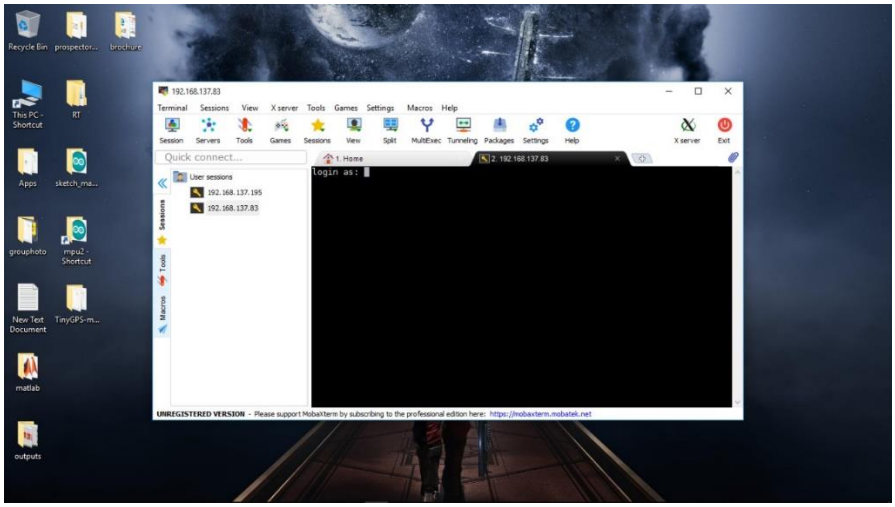


We use SSH (Secure Shell) for operating network services securely over an unsecured network.

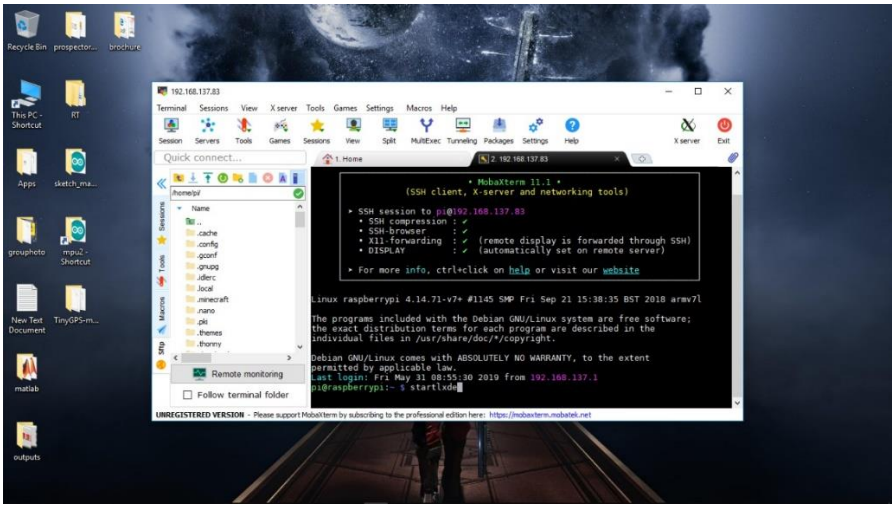


As shown in the above figure we will use the IP address of Pi3(Previously Found) in the Remote host or Network name can be used.

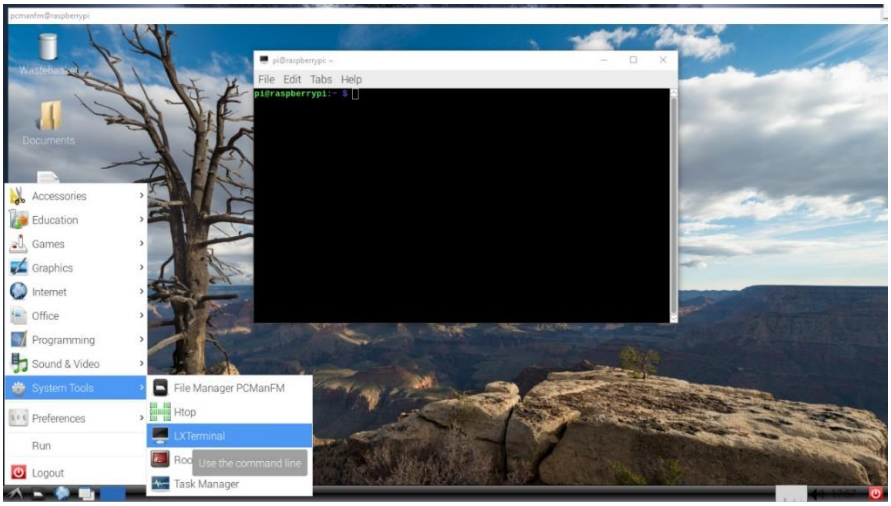
Step5: Now as the Pi3 is configured, we login into our Pi3 using login credentials synch as user id and password.



We use the command ***$ startlxde*** to open the virtual screen.



Step6: We use LX terminal which Is similar to command prompt of Windows.



Step7: Then, installation of all the required packages will be done

Sensors-Libraries

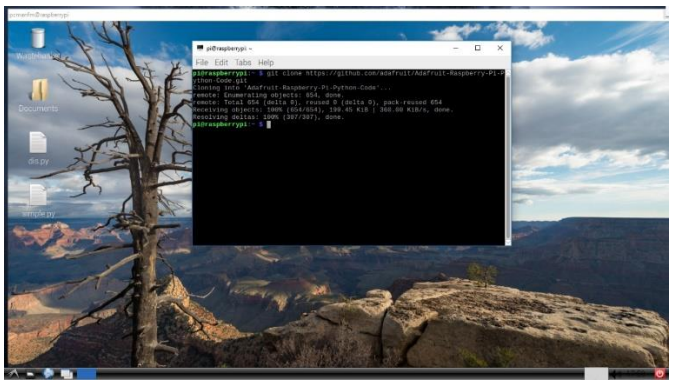
DHT11- Adafruit_Python_DHT

LCD-RPLCD

BMP180-Smbus (for I2C) protocol and Adafruit_BMP085

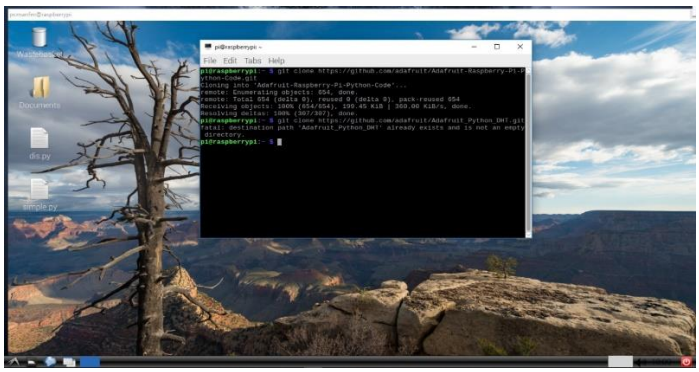
For BMP180 and DHT11 the package will be installed using GitHub so we need to use the commands respectively.

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



Followed by the Command

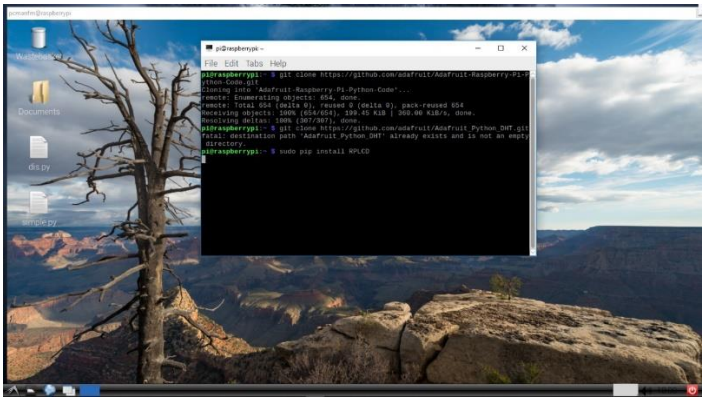
$git clone https://github.com/Adafruit_Python_DHT.git



For LCD module:

We use the following command for LCD:

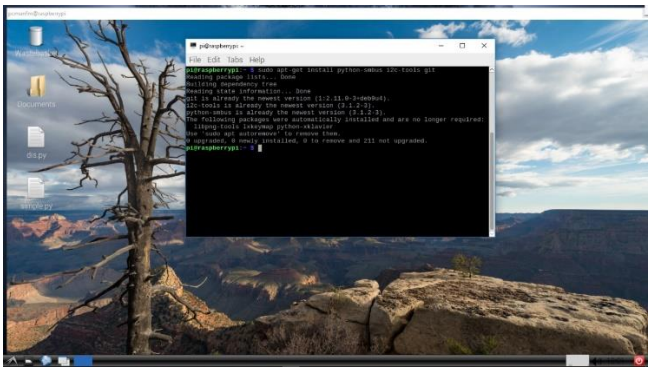
$sudo pip install RPLCD



For BMP180 I2C bus Interface:

We use the command

$sudo apt-get install python-smbus I2c-tools git



To check the BMP085, following codes are executed:

$ cd Adafruit-Raspberry-Pi-Python-Code

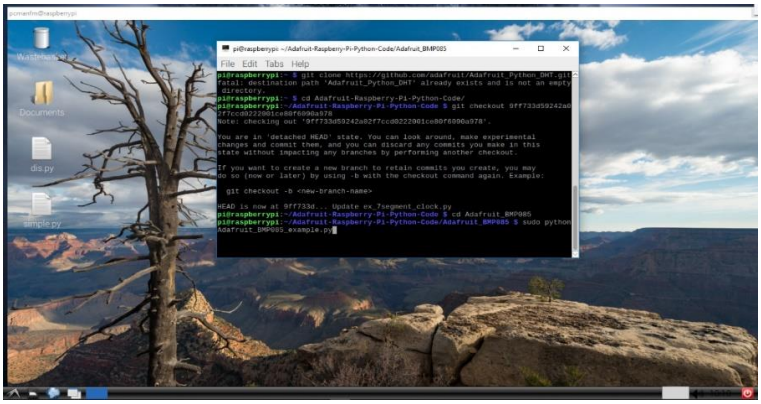
~/ Adafruit-Raspberry-Pi-Python-Code

$ cd Adafruit_BMP085

~/ Adafruit-Raspberry-Pi-Python-Code/Adafruit_BMP085

$sudo python Adafruit_BMP085_example.py

We will get values of Sensor as Output.



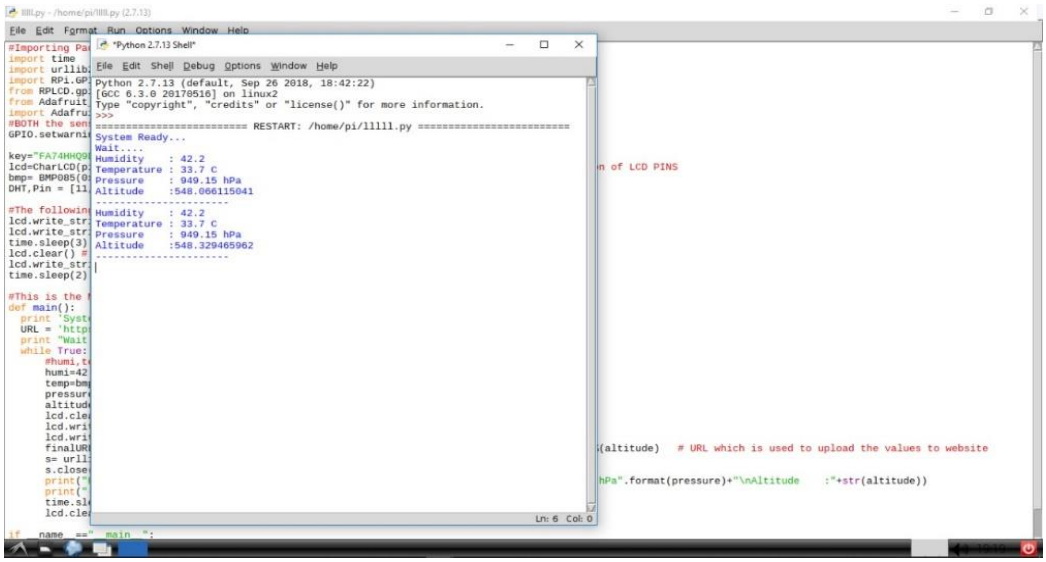
9.2. Code and Output:

The Main Code of the Project is as follows:

```
#Importing Packages required to run a Program
import time
import urllib2      #This module is for Thingspeak(URL Handling Module)
import RPi.GPIO as GPIO
from RPLCD.gpio import CharLCD          # This module is for LCD(16X2)
from Adafruit_BMP085 import BMP085      # This module is for Pressure Sensor(BMP180)
```

```
import Adafruit_DHT       # This module is for Humidity Sensor(DHT11)
#BOTH the sensors can measure Temperature
GPIO.setwarnings(False)

key="FA74HHQ9DB649OJ3"      # Type your Write API key from ThingSpeak
lcd=CharLCD(pin_rs=12,pin_e=16,pins_data=[18,36,38,40],numbering_mode=GPIO.BOARD) # Initialisation of
LCD PINS
bmp= BMP085(0x77) # Initialisation of BMP085(Port)
DHT,Pin = [11,4] # Initialisation of DHT11 pins (here 4 is GPIO4)

#The following lines will be displayed on LCD with certain delays
lcd.write_string("ECIL")
lcd.write_string("\n\rWelcomes you")
time.sleep(3)
lcd.clear() # Clears the text on LCD
lcd.write_string("RPI  Weather \n\rMonitoring")
time.sleep(2)

#This is the Main function executes continuously
def main():
  print 'System Ready...'
  URL = 'https://api.thingspeak.com/update?api_key=%s' % key
  print "Wait...."
  while True:
     humi,temp= Adafruit_DHT.read_retry(DHT,Pin)
     temp=bmp.readTemperature()
     pressure=bmp.readPressure()/100.0
     altitude=bmp.readAltitude()
     lcd.clear()
     lcd.write_string("Humi#Temp#P(hPa)")
     lcd.write_string("\n\r%s" %humi+" %sC %s" %(temp, pressure))
     finalURL = URL+"&field1=%s&field2=%s" %(humi, temp)+"&field3=%s" %(pressure)+"&field4=%s"
%(altitude)   # URL which is used to upload the values to website
     s= urllib2.urlopen(finalURL);  #  This command opens the URL for fetching data
     s.close() # Closing the URL
     print("Humidity    : "+str(humi)+"\nTemperature : {0} C".format(temp)+"\nPressure    : {0}
hPa".format(pressure)+"\nAltitude    :"+str(altitude))
     print("----------------------")
     time.sleep(7)
     lcd.clear()

if __name__=="__main__":
   main()
```

OUTPUTS:


Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (default, Sep 26 2018, 18:42:22)
[GCC 6.3.0 20170516] on linux2
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: /home/pi/llllll.py ==========
System Ready...
Wait....
Humidity : 42.2
Temperature : 33.7 C
Pressure : 949.15 hPa
Altitude :548.066115041

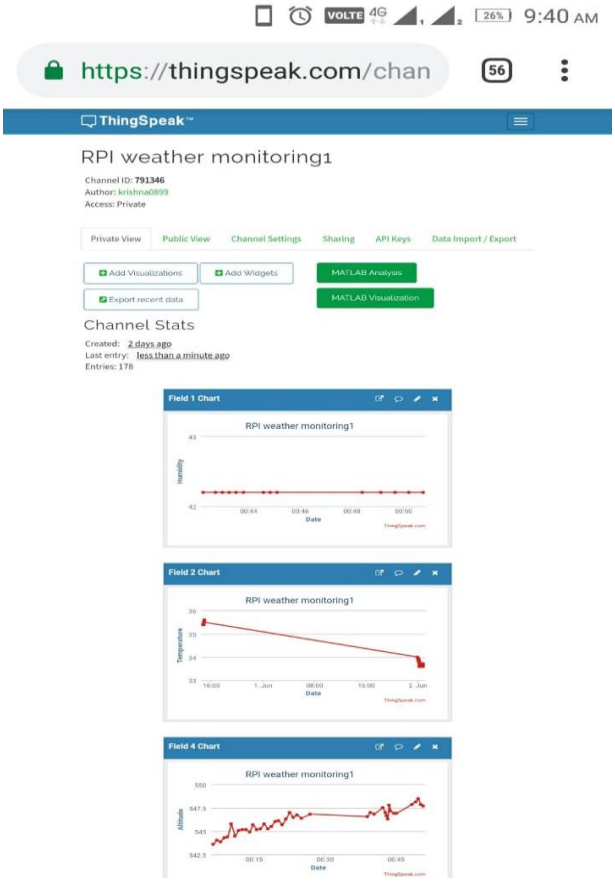
Humidity : 42.2
Temperature : 33.7 C
Pressure : 949.15 hPa
Altitude :548.329465962


ECIL
Welcomes you


RPI Weather
Monitoring


Humi#Temp#P(hPa)
42.2 33.7C 949.1

The Output of the ThingsSpeak will be as follows:



Conclusion and Future Scope

Since many years, various research operations are being performed in the field of weather monitoring. One such good attempt is this i.e. to monitor the weather parameters in one locality and to share the data globally through cloud. Thus, in future modifications can be made on this system to make it serve for other applications too. We can add the new features like -

- Weather prediction is a very important factor, which forecasts the climate in a region based upon the values of weather parameters. So, the calculated results from this system can be made use in forecasting the weather of that locality for a period of time.

- As we made use of Raspberry PI3 in this model, immediate alert message or e-mail can be sent to the mobile phone by implementing an app which supports the android and other operating systems.

So, that we can check the data from anywhere at any time by using the internet. It is very easy to install the app and check the data whenever we want. This will be more beneficial for everyone as in every home there is at least one smart phone in these days.

• As the applications are limitless, other weather parameters can also be monitored easily with the addition of related sensors to the system architecture. Such as, by including the sensors of soil moisture, PH values, Air Direction and others, we can use this in agricultural fields. So, that it would be helpful to farmers to take care of crop yield.

Bibliography

- Design & Development of IoT Application Using Raspberry Pi3 & ThingSpeak A Practical Approach-Microsoft
- IoT Based Weather Monitoring System using Raspberry Pi-IRJET
- DHT 11 Humidity & Temperature Sensor -Mouser Electronics
- BMP180 Digital pressure sensor-BOSCH
- 16x2 LCD-Engineers Garage
- RASPBERRY PI 3 MODEL B-Farnell and Newark
- Introduction to Raspberry Pi with Raspbian OS-Guruprasad . K. Basavaraju

Web References:

- https://www.raspberrypi-spy.co.uk/2015/06/basic-temperature-logging-to-the-internet-with-raspberry-pi/
- https://tutorials-raspberrypi.com/raspberry-pi-and-i2c-air-pressure-sensor-bmp180/
- https://circuitdigest.com/microcontroller-projects/iot-temperature-humidity-monitoring-using-arduino/
- https://youtu.be/DPvxsHoD7kchttps://youtu.be/DPvxsHoD7kc
- http://www.circuitbasics.com/raspberry-pi-lcd-set-up-and-programming-in-python
- https://www.hackster.io/weargenius/bmp180-interfacing-with-raspberry-pi-in-detail-af06a8
- https://github.com/github