ABSTRACT

Everything has been digitized. In many existing systems, almost all petrol pumps have a controlling unit to perform the tasks like managing the electrical pump, drive the display, measure the flow & accordingly turn OFF the electrical pump. But still a person is required to collect the money and there is a possibility of many human errors. In this proposed petrol pump automation system, we are using RFID card to access petrol at different petrol stations of different petrol providing companies across the country and here we are connecting all these petrol stations using single web server. This web server access is secured by a password which is known only to the petrol companies. Whenever we want to fill the tank from the fuel dispenser, we just have to place the RFID card near the RFID reader. Then the microcontroller reads the data from the RFID reader and performs the action according to the customer requirements. This digital petrol pump system also provides the security for the customers for filling petrol at the Petrol stations by avoiding the involvement of human beings, hence reduces the risk of carrying money every time. This petrol pump system consists of Raspberry-pi Development Board, RFID module, LCD display, Keypad and Ac pump. When RFID reader reads the card, the system asks for the amount. On entering the amount, the motor starts and petrol gets filled in the petrol tank from the fuel dispenser then it also shows the remaining balance amount.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

What is an embedded system?

An Embedded System is a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a specific function. An embedded system is a microcontroller-based, software driven, reliable, real-time control system, autonomous, or human or network interactive, operating on diverse physical variables and in diverse environments and sold into a competitive and cost conscious market.

An embedded system is not a computer system that is used primarily for processing, not a software system on PC or UNIX, not a traditional business or scientific application. High-end embedded & lower end embedded systems. High-end embedded system - Generally 32, 64 Bit Controllers used with OS. Examples Personal Digital Assistant and Mobile phones etc. Lower end embedded systems - Generally 8, 16 Bit Controllers used with a minimal operating systems and hardware layout designed for the specific purpose. Examples Small controllers and devices in our everyday life like Washing Machine, Microwave Ovens, where they are embedded in

What is IOT?

The Internet of Things (IoT) refers to a system of interrelated, internet-connected objects that are able to collect and transfer data over a wireless network without human intervention.

The personal or business possibilities are endless. A ‘thing’ can refer to a connected medical device, a biochip transponder (think livestock), a solar panel, a connected automobile with sensors that alert the driver to a myriad of possible issues (fuel, tire pressure, needed maintenance, and more) or any object, outfitted with sensors, that has the ability to gather and transfer data over a network.

Today, businesses are motivated by IoT and the prospects of increasing revenue, reducing operating costs, and improving efficiencies. Businesses also are driven by a need for regulatory compliance. Regardless of the reasons, IoT device deployments provide the data and insights necessary to streamline workflows, visualize usage patterns, automate processes, meet compliance requirements, and compete more effectively in a changing business environment.

OBJECTIVE

This IOT based Petrol Pump is created keeping four main objectives in mind,

i.e.-

1. Digitalization
2. Centralization
3. Reduction of Transportation of Cash
4. Reduction of Labor Cost

Digitalization

Digitalization is the use of digital technologies to change the business model and provide new revenue and value-producing opportunities; it is the process of moving to a digital business.

In this "IOT based Petrol Pump" every single operation is digitalized like customer can add the balance by going to the web server, check balance and also check the previous amounts of fuel they bought. Same for the petrol providing companies, they can check all the transactions done by the customers and the available balance of the customers.

Centralization

Centralization refers to the process in which all the data of an organization will be stored at a specific location and that data is only accessible by the higher authorities of that organization and to the specific people whom the authorization is given.

In this project every data of the customers will be stored at a centralized database and that can be accessible by the customers and the petrol providing companies. By implementing this process human effort and interaction with the customer data will be a way lower which will further reflect in lower possibility of data breach and data input will be accurate.

Reduction of Transportation of Cash

By implementing the centralization process in this project helps us reducing the cash transaction which reflects in proper tax filing of the petrol proving companies and the reduction of black money.

In this process a centralized wallet will be there in which the added money of customers will be stored and when the customer will buy the fuel the same amount will be deducted from the customer's wallet and will be added to the wallet of petrol proving company's wallet.

Reduction of Labor Cost

Every petrol pump has employees just to provide petrol to the customers and keep the cash, in this project we have developed a very simple user interface which the customer can access very easily on their own. But keeping the employment problem of India in mind we can reduce some employees who are employed to provide fuel and in order to maintain and check the technicalities of system we will need to appoint a technical supervisor for every petrol pump.

The reduction of the unskilled employees will force the uneducated people to be educated; this also will solve another major problem of the country which is illiteracy. And the reduction will also help the petrol pump owners to cut down their commission to some extend which will be also a profit for the customers as the fuel price will get lower.

ADVANTAGES OF THE SYSTEM

- It saves the precious time
- It is easy to use for both customers and fuel proving companies because of the very simple user interface
- When the customer will fill fuel on their own, there will be no chance of cheating
- It provides data accuracy which makes it reliable
- It is accessible from every petrol pump, all over India
- The information of customers are kept secured as one customer can only login via their own user id and password
- Customer can get information from a single web server which makes it centralized
- It reduces the cost labor which reflects in reduced fuel price
- It increases the proper tax filing which also reduces the black money
- It will help increase literacy of our country as a lot of skilled workers are required to maintain this system

CHAPTER 2

SOFTWARE AND HARDWARE REQUIREMENTS

Softwares Required

- Boot loader: NOOBS
- Raspberry-Pi O.S: Raspbian
- Programming Language: Python
- IDE: Python3(IDLE or Any Python IDE)
- Brower: (Google Chrome or any Browser)
- IOT Server

Hardware Components Required

- Raspberry Pi
- Extender PCB Board
- RFID Module
- RFID Cards
- LCD Display
- Keypad
- Submersible Pump
- 40 pin Connector
- Dc Power Jack
- DC Power Adapter
- Relay Switch
- LED
- 6amp Fuse
- Resistor
- Capacitor
- Diode
- Transistor
- Cable and Connector

Hardware Description

4.5 RASPBERRY PI 3 (MODEL B)

The Raspberry Pi 3 is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

The Raspberry Pi 3 is the third generation Raspberry Pi. It replaced the Raspberry Pi 2 Model B in February 2016. Compared to the Raspberry Pi 2 it has:

- A 1.2GHz 64-bit quad-core ARMv8 CPU
- 802.11n Wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE

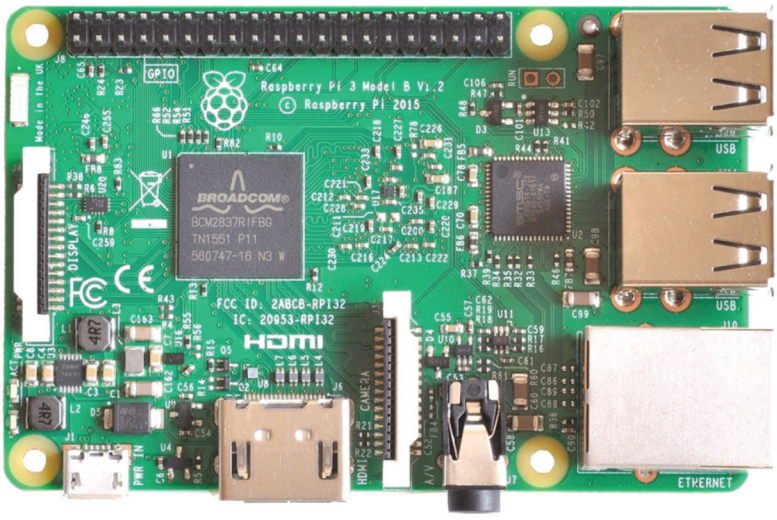


FIG 4.5(A): RASPBERRY PI 3

Raspberry Pi 3 V/S Pi 2: Hardware

CPU:

The Raspberry Pi 2 has a quad-core 900MHz CPU, the Pi 3 a quad-core 1.2GHz one. Both have 1GB RAM and both use a fourth-generation VideoCore CPU. The performance boost is very big. First, while the CPU only gains 300MHz, it also updates its architecture from a Cortex-A7 set to a Cortex-A53 one. This is an architecture boost from 32-bit to 64-bit, and gets you better performance for your clock speed.

GPU:

Both are equipped with VideoCore IV chipsets, but where the Pi 2 is clocked at 250MHz, this new one is 400MHz. While they're both designed to deliver 1080p video rather than 4K, this means the graphics chip has been scaled up with the CPU.

RAM:

Both models have 1GB, both are forms of DDR2. But where the Raspberry Pi 2 has 450MHz, RAM, the Pi 3 has 900MHz RAM

Features:

Like the Pi 2, it also has:

- 1GB RAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot (now push-pull rather than push-push)
- VideoCore IV 3D graphics core

The Raspberry Pi 3 has an identical form factor to the previous Pi 2 (and Pi 1 Model B+) and has complete compatibility with Raspberry Pi 1 and 2.

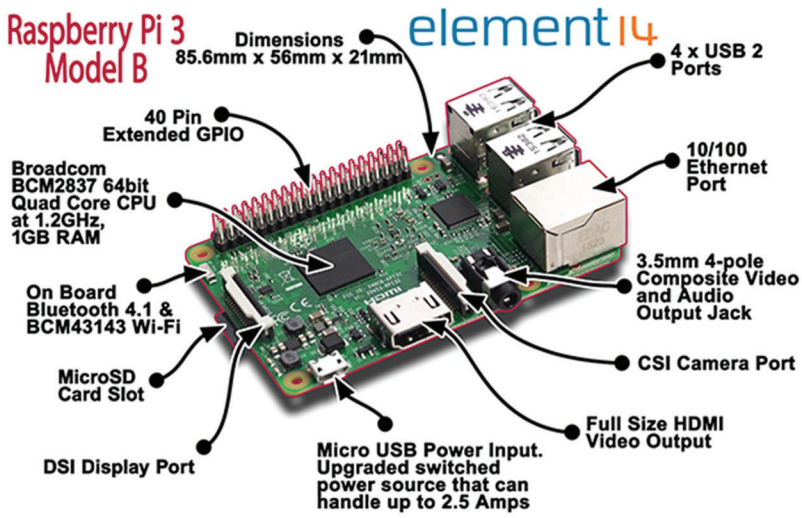


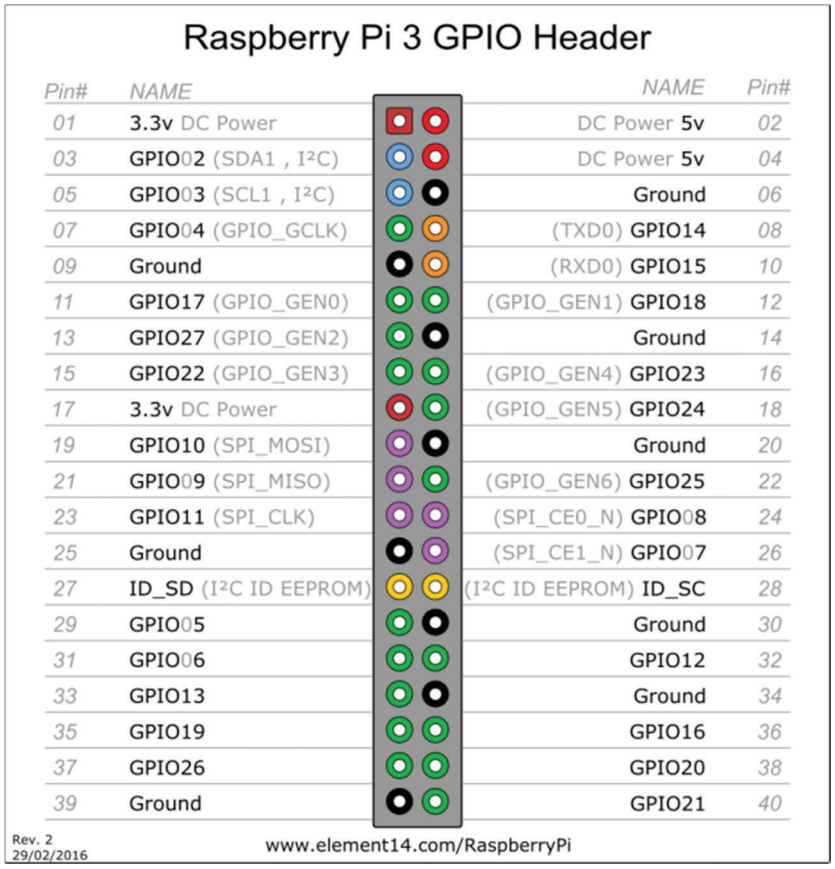
FIG 4.5(B): RASPBERRY PI 3 MODULE DESCRIPTION

Raspberry Pi 3 technical specifications:

- SoC – Broadcom BCM2837 64bit ARMv8 quad core Cortex A53 processor @ 1.2GHz with dual core VideoCore IV GPU @ 400 MHz supporting OpenGL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode. Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
- System Memory – 1GB LPDDR2
- Storage – micro SD slot
- Video & Audio Output – HDMI 1.4 and 4-pole stereo audio and composite video port
- Connectivity – 10/100M Ethernet, WiFi 802.11 b/g/n up to 150Mbps and Bluetooth 4.1 LE (BCM43438 module)
- USB – 4x USB 2.0 host ports (with better power management, allowing higher power peripherals), 1x micro USB port for power

- Expansion
    - 40-pin GPIO header
    - MIPI DSI for Raspberry Pi touch screen display
    - MIPI CSI for Raspberry Pi camera
- Power Supply – 5V up to 2.4A via micro USB port
- Dimensions – 85 x 56 x 17 mm

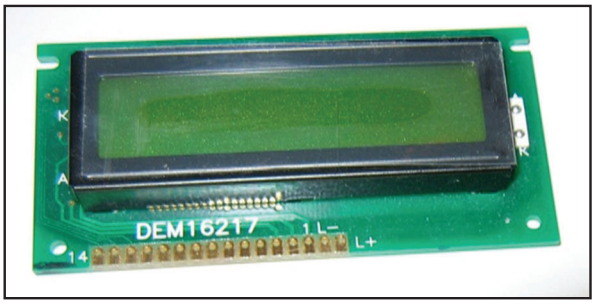
GPIO Pins of Raspberry Pi 3



LCD Display

Frequently, an 8051 program must interact with the outside world using input and output devices that communicate directly with a human being. One of the most common devices attached to an 8051 is an LCD display. Some of the most common LCDs connected to the 8051 are 16x2 and 20x2 displays. This means 16 characters per line by 2 lines and 20 characters per line by 2 lines, respectively.

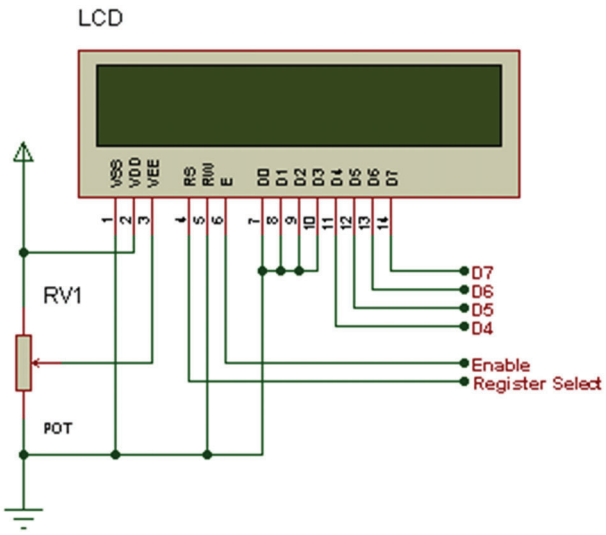
Fortunately, a very popular standard exists which allows us to communicate with the vast majority of LCDs regardless of their manufacturer. The standard is referred to as HD44780U, which refers to the controller chip which receives data from an external source (in this case, the 8051) and communicates directly with the LCD.



LCD Display

44780 LCD BACKGROUND

The 44780 standard requires 3 control lines as well as either 4 or 8 I/O lines for the data bus. The user may select whether the LCD is to operate with a 4-bit data bus or an 8-bit data bus. If a 4-bit data bus is used the LCD will require a total of 7 data lines (3 control lines plus the 4 lines for the data bus). If an 8-bit data bus is used the LCD will require a total of 11 data lines (3 control lines plus the 8 lines for the data bus).



The three control lines are referred to as EN, RS, and RW.

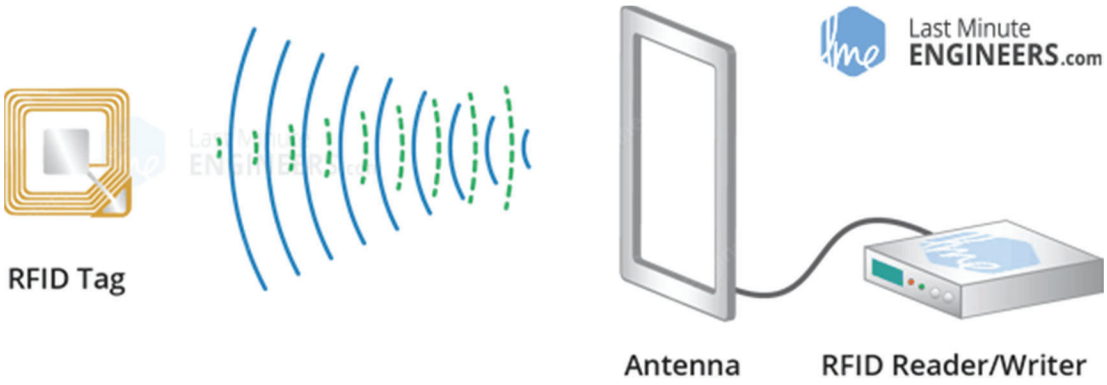
The EN line is called "Enable." This control line is used to tell the LCD that you are sending it data. To send data to the LCD, your program should make sure this line is low (0) and then set the other two control lines and/or put data on the data bus. When the other lines are completely ready, bring EN high (1) and wait for the minimum amount of time required by the LCD datasheet (this varies from LCD to LCD), and end by bringing it low (0) again.

The RS line is the "Register Select" line. When RS is low (0), the data is to be treated as a command or special instruction (such as clear screen, position cursor, etc.). When RS is high (1), the data being sent is text data which should be displayed on the screen. For example, to display the letter "T" on the screen you would set RS high.

The RW line is the "Read/Write" control line. When RW is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively querying (or reading) the LCD. Only one instruction ("Get LCD status") is a read command. All others are write commands--so RW will almost always be low .Finally, the data bus consists of 4 or 8 lines (depending on the mode of operation selected by the user). In the case of an 8-bit data bus, the lines are referred to as DB0, DB1, DB2, DB3, DB4, DB5, DB6, and DB7.

RFID Module

RFID or Radio Frequency Identification system consists of two main components, a transponder/tag attached to an object to be identified, and a Transceiver also known as interrogator/Reader.



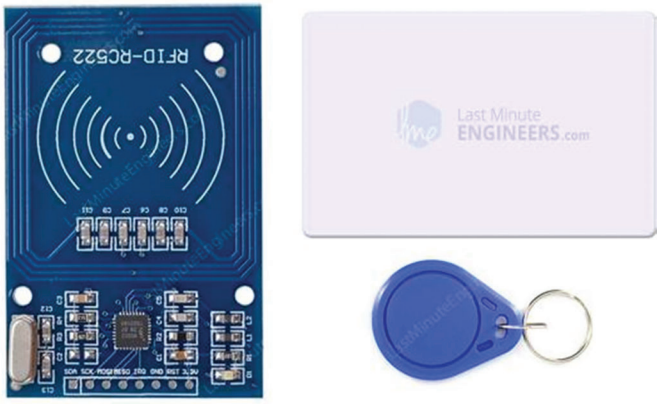
A Reader consists of a Radio Frequency module and an antenna which generates high frequency electromagnetic field. On the other hand, the tag is usually a passive device, meaning it doesn't contain a battery. Instead it contains a microchip that stores and processes information, and an antenna to receive and transmit a signal.

To read the information encoded on a tag, it is placed in close proximity to the Reader (does not need to be within direct line-of-sight of the reader). A Reader generates an electromagnetic field which causes electrons to move through the tag's antenna and subsequently power the chip.

The powered chip inside the tag then responds by sending its stored information back to the reader in the form of another radio signal. This is called backscatter. The backscatter, or change in the electromagnetic/RF wave, is detected and interpreted by the reader which then sends the data out to a computer or microcontroller.

Hardware Overview – RC522 RFID Reader/Writer Module

The RC522 RFID module based on MFRC522 IC from NXP is one of the most inexpensive RFID options that you can get online for less than four dollars. It usually comes with a RFID card tag and key fob tag having 1KB memory. And best of all, it can write a tag, so you can store your some sort of secret message in it.



The RC522 RFID Reader module is designed to create a 13.56MHz electromagnetic field that it uses to communicate with the RFID tags (ISO 14443A standard tags). The reader can communicate with a microcontroller over a 4-pin Serial Peripheral Interface (SPI) with a maximum data rate of 10Mbps. It also supports communication over I2C and UART protocols.

The module comes with an interrupt pin. It is handy because instead of constantly asking the RFID module "is there a card in view yet? The module will alert us when a tag comes into its vicinity.

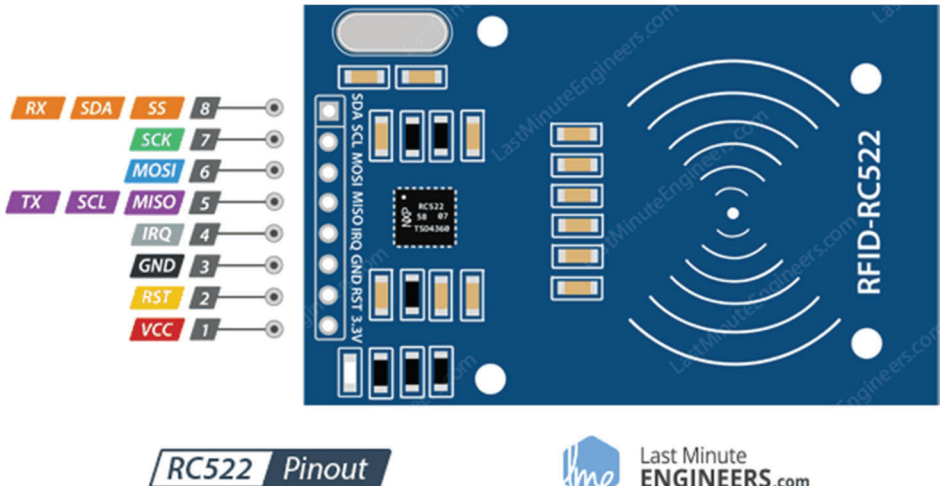
The operating voltage of the module is from 2.5 to 3.3V, but the good news is that the logic pins are 5-volt tolerant, so we can easily connect it to a Raspberry-pi or any 5V logic microcontroller without using any logic level converter.

Here are complete specifications:

Frequency Range	13.56 MHz ISM Band
Host Interface	SPI / I2C / UART
Operating Supply Voltage	2.5 V to 3.3 V
Max. Operating Current	13-26mA
Min. Current(Power down)	10µA
Logic Inputs	5V Tolerant
Read Range	5 cm

RC522 RFID Module Pinout

The RC522 module has total 8 pins that interface it to the outside world. The connections are as follows:



VCC supplies power for the module. This can be anywhere from 2.5 to 3.3 volts. You can connect it to 3.3V output from your Arduino. Remember connecting it to 5V pin will likely destroy your module!

RST is an input for Reset and power-down. When this pin goes low, hard power-down is enabled. This turns off all internal current sinks including the oscillator and the input pins are disconnected from the outside world. On the rising edge, the module is reset.

GND is the Ground Pin and needs to be connected to GND pin on the Arduino.

IRQ is an interrupt pin that can alert the microcontroller when RFID tag comes into its vicinity.

MISO / SCL / Tx pin acts as Master-In-Slave-Out when SPI interface is enabled, acts as serial clock when I2C interface is enabled and acts as serial data output when UART interface is enabled.

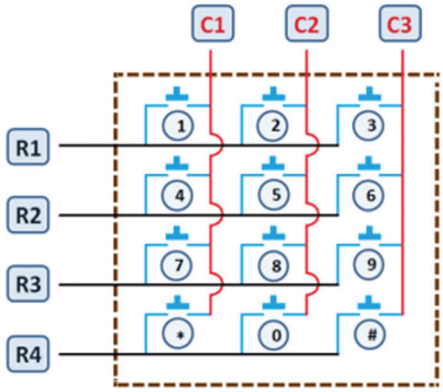
MOSI (Master Out Slave In) is SPI input to the RC522 module.

SCK (Serial Clock) accepts clock pulses provided by the SPI bus Master i.e. Arduino.

SS / SDA / Rx pin acts as Signal input when SPI interface is enabled, acts as serial data when I2C interface is enabled and acts as serial data input when UART interface is enabled. This pin is usually marked by encasing the pin in a square so it can be used as a reference for identifying the other pins.

Keypad

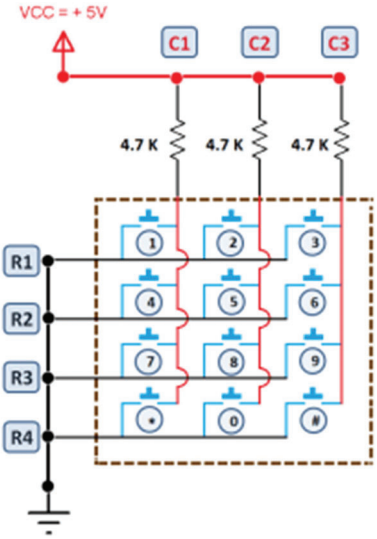
Keypad/Keyboard is most widely used as an input device. Generally Used with microcontroller and microprocessor based devices. In this section, we will discuss the **"4×3" matrix keypad**, its basic understanding is important. Here basic refers to its construction and working mechanism of key detection. Here R1, R2, R3 & R4 refers to row 1, row 2, row 3 & row 4 respectively and C1, C2 & C3 refers to column 1, column 2 & column 3 respectively.



OPERATING MECHANISM OF KEYPAD 4X3 | BUTTON MATRIX CONCEPT

We can see clearly there are 12 numbers and/or characters and 12 push-buttons (switch). Each push-button is associated with each number or character.

All the columns are set to HIGH i.e. + 5V and rows to LOW i.e. ground (we can interchange columns to LOW and rows to HIGH according to our convenience, but then we need to change the programming model accordingly).



So if **no key** has been pressed then all columns will remain HIGH and if the **key** has been pressed then its corresponding column will give a LOW signal (because it shorted to ground via row). Just suppose we press '2', it is in column 2 and row 1 so as we pressed '2' column 2 will become from HIGH to LOW (because signal directs to the ground without any resistance) but as we release the key 2 it will again become high because of **pull-up resistor**, we read about the concept of pull-up in the previous chapter, here we used pull-up resistor of 4.7 K ohm.

LOGIC BEHIND KEY DETECTION OF KEYPAD 4×3

First let us understand the logic to **detect the key and print numbers** of the First **row**, as now we have **three columns** and **one row**. So there are three possibilities

1. **Column 1 key pressed**
2. **Column 2 key pressed**
3. **Column 3 key pressed**

If column 1 key has been pressed, as row R1 is output type and C1 is input to Arduino, then Arduino will read the logic (state HIGH or LOW) of Row 1 passes through C1 line (same logic we used in reading input from a switch (interface of the switch as input)).

Row 1 should be **LOW,** so that the microcontroller can detect logic (as we know from both sides HIGH logic will not lead to a potential difference). But if we have multiple rows then we should give logic **HIGH to them (R2, R3, R4)** to distinguish between R1 and other rows. In short for Row 1 keys

1. **R1 → LOW**
2. **R2 → HIGH**
3. **R3 → HIGH**
4. **R4 → HIGH**
5.

Raspbery-pi will continuously read input pin signals, if C1 is LOW means key 1 is pressed, if C2 is LOW means key 2 is pressed and if C3 is LOW means key 3 is pressed.

A. If **R1 → LOW**, R2 → HIGH, R3 → HIGH and R4 → HIGH
Column 1 key pressed leads to detect '1'
Column 2 key pressed leads to detect '2'
Column 3 key pressed leads to detect '3'

Similarly,

B. If R1 → HIGH, **R2 → LOW**, R3 → HIGH and R4 → HIGH
Column 1 key pressed leads to detect '4'
Column 2 key pressed leads to detect '5'
Column 3 key pressed leads to detect '6'

C. If R1 → HIGH, R2 → HIGH, **R3 → LOW** and R4 → HIGH
Column 1 key pressed leads to detect '7'
Column 2 key pressed leads to detect '8'
Column 3 key pressed leads to detect '9'

D. If R1 → HIGH and R2 → HIGH, R3 → HIGH and **R4 → LOW**
Column 1 key pressed leads to detect '*'
Column 2 key pressed leads to detect '0'
Column 3 key pressed leads to detect '#'

DC Submersible Pump

Submersible pumps are the pumping machines that are designed to submerge into liquids. The DC submersible pump refers to the submersible pumps that use the direct current electricity as the power source.

The DC submersible pumps as well as other types of submersible pumps look like a metal tube and their goal is to push the water to the surface. This type of pipe-look water pumps conceals the motor within the tube so that the liquid outside the pump does not get access to the electrical parts inside the pump.

The DC submersible pumps are versatile and can better accommodate with different working environments because they can be powered from a variety of independent DC power sources. The most common power sources for the DC submersible pumps are the solar modules, batteries, or generators. The DC submersible pumps with solar power supplies sometimes have a controller in order to enhance the current when the sunlight is low.

More Information on the Power Supply

Typically, the DC submersible pumps can operate on multiple volts of DC power such as 6V, 12V, 24V or 32V. There are some advantages of applying the direct current instead of the alternating current (AC). The first advantage is that the DC power system allows the submersible pumps to adapt to more power supply options such as batteries. It makes the machines portable and is more convenient to apply.

Also, the AC power system needs a controller to control the speed while the AC power system does not need such device. In most cases, the controller of the power system is only needed when the solar power modules are applied.

The DC submersible pumps are more energy efficient too; it takes less power to run a DC submersible pump than an AC model. However, there are limits to the DC machines as well. The DC submersible pumps have a shorter lifespan than the AC counterparts. Besides, though the DC machines are more energy efficient and easier to operate without a current controller, they tend to work in a rather lower speed and they are less powerful in pumping.

The Application of Submersible Pumps

The submersible pumps are applicable in multiple areas. For example, they can be used to pump sewage from the septic tanks and it is convenient because the pump deliver the sewage directly to tanks so that they can be transferred to treatment facilities and handled later on.

Secondly, they can be used both industrially and in buildings. Wherever there is too much unwanted water in a work site or when the basement of a building is flooded, the submersible pump can be applied to extract all the water out of the place easily.

The other common use of a submersible pump is for the oil wells. Whether it is an inland oil well or an offshore oil well, the submersible pump can pump out the oil from the ground and transfer it to the factory for further process. Some types of submersible pumps can also be used for drilling wells to a certain depth.

They are called the drilling pumps or borehole pumps. In addition to the industrial use, the submersible pumps can be used agriculturally as a part of the irrigation systems.

What Are the Advantages of a Submersible Pump?

The main goal of a submersible pump is to pump out liquids from the ground or from places under the surface level. The submersible pumps are best for job because of several reasons. For one thing, the submersible pump keeps almost all the components within the pipe-like body; therefore, it does not take up much space.
In other words, the submersible pumps can work best in a confined environment comparing to other types of pumps. The submersible pumps connect directly to the storage tanks with pipes or hoses which also take up very limited space as well. As a result, they accommodate with most working environments possible.

The other quality that the submersible pumps have which again comes from the design is that it is the safest options to operate among all other types of water pumps. Since all the working parts of the machine is contained within the body and the whole body will be in the pit of the work site, there is not much of a chance that the user around the machine gets hurt when it is running. This feature alone makes the submersible pumps stand out and become widely accepted.

Once again, because all the components are carried inside the pump body and they operate under the water in most cases, they do not make loud noises. Or to be specific, the noises made while the machine is running are all blocked by the water and the case of the machine. What's more, the simplicity of the mechanism makes this kind of water pump more effective when it comes to pumping out liquids from underground.

In addition to the safety, the efficiency and the quietness a submersible pump have, the maintenance of the machine is also quite easy. In fact, it does not require much maintenance because the working parts and electrical components are all protected inside the case of the machine, the small contaminants are not likely to enter the body and harm the machine.
Moreover, because the submersible pumps are working in water, they are cooled by the water when running and the machine can last longer with this style of operation.

The application of the DC submersible pumps is mainly in the so called dewatering operations and because of the design and the power supply that they work with, there are a lot of advantages to the use of them.

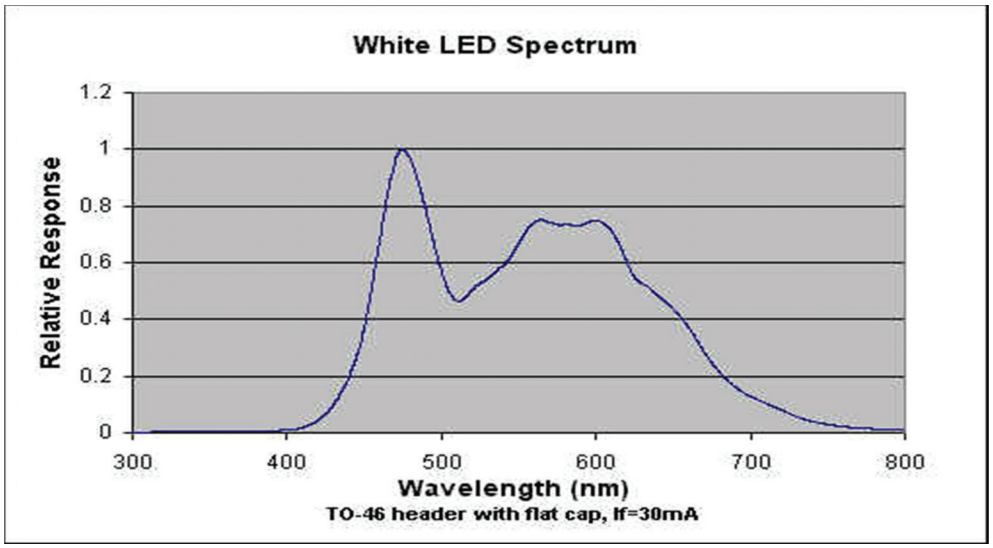
The primary advantage of a DC submersible water pump is its convenience and ease of operation because the mechanism of the pump is simple and the power system allows more portability to the machine. If the users are looking for a reliable yet compact water pump with multiple power supply options, the DC submersible pump may just be the right choice for them.

LED

Light Emitting Diodes (LED) have recently become available that are white and bright, so bright that they seriously compete with incandescent lamps in lighting applications. They are still pretty expensive as compared to a GOW lamp but draw much less current and project a fairly well focused beam.

The diode in the photo came with a neat little reflector that tends to sharpen the beam a little but doesn't seem to add much to the overall intensity.

When run within their ratings, they are more reliable than lamps as well. Red LEDs are now being used in automotive and truck tail lights and in red traffic signal lights. You will be able to detect them because they look like an array of point sources and they go on and off instantly as compared to conventional incandescent lamps.



LEDs are monochromatic (one color) devices. The color is determined by the band gap of the semiconductor used to make them. Red, green, yellow and blue LEDs are fairly common. White light contains all colors and cannot be directly created by a single LED. The most common form of "white" LED really isn't white. It is a Gallium Nitride blue LED coated with a phosphor that, when excited by the blue LED light, emits a broad range spectrum that in addition to the blue emission, makes a fairly white light.

There is a claim that these white LED's have a limited life. After 1000 hours or so of operation, they tend to yellow and dim to some extent. Running the LEDs at more than their rated current will certainly accelerate this process.

There are two primary ways of producing high intensity white-light using LED'S. One is to use individual LED'S that emit three primary colours—red, green, and blue—and then mix all the colours to form white light. The other is to use a phosphor material to convert monochromatic light from a blue or UV LED to broad-spectrum white light, much in the same way a fluorescent light bulb works. Due to metamerism, it is possible to have quite different spectra that appear white.

LEDs are semiconductor devices. Like transistors, and other diodes, LEDs are made out of silicon. What makes an LED give off light are the small amounts of chemical impurities that are added to the silicon, such as gallium, arsenide, indium, and nitride.

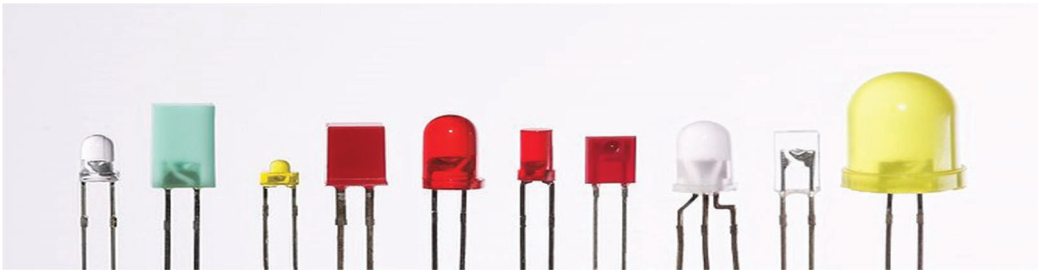
When current passes through the LED, it emits photons as a byproduct. Normal light bulbs produce light by heating a metal filament until it is white hot. LEDs produce photons directly and not via heat, they are far more efficient than incandescent bulbs.

Fig 4.9(a): circuit symbol

Not long ago LEDs were only bright enough to be used as indicators on dashboards or electronic equipment. But recent advances have made LEDs bright enough to rival traditional lighting technologies. Modern LEDs can replace incandescent bulbs in almost any application.

Types of LED'S

LEDs are produced in an array of shapes and sizes. The 5 mm cylindrical package is the most common, estimated at 80% of world production. The color of the plastic lens is often the same as the actual color of light emitted, but not always. For instance, purple plastic is often used for infrared LEDs, and most blue devices have clear housings. There are also LEDs in extremely tiny packages, such as those found on blinkers and on cell phone keypads. The main types of LEDs are miniature, high power devices and custom designs such as alphanumeric or multi-color.



Different types of LED'S

Relay Switch

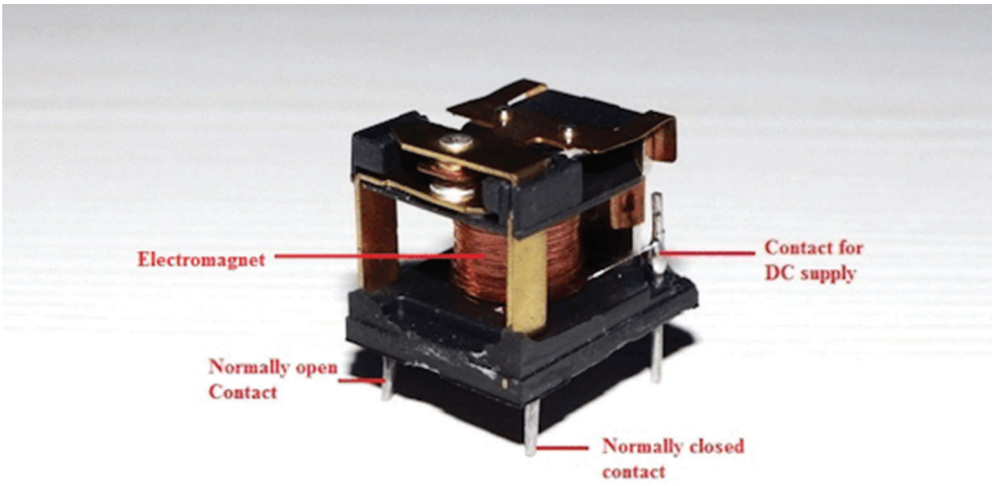
A relay can be defined as a switch. Switches are generally used to close or open the circuit manually .Relay is also a switch that connects or disconnects two circuits. But instead of manual operation a relay is applied with electrical signal, which in turn connects or disconnects another circuit.

Relays can be of different types like electromechanical, solid state. Electromechanical relays are frequently used. Let us see the internal parts of this relay before knowing about it working. Although many different types of relay were present, their working is same.

Every electromechanical relay consists of an consists of an

1. Electromagnet
2. Mechanically movable contact
3. Switching points and
4. Spring

Electromagnet is constructed by wounding a copper coil on a metal core. The two ends of the coil are connected to two pins of the relay as shown. These two are used as DC supply pins.



Generally two more contacts will be present, called as switching points to connect high ampere load. Another contact called common contact is present in order to connect the switching points.

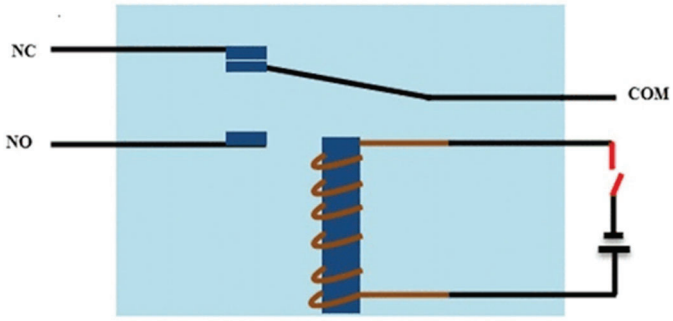
These contacts are named as normally open (NO), normally closed (NC) and common (COM) contacts.

Relay can be operated using either AC or DC.

In case of AC relays, for every current zero position, the relay coil gets demagnetized and hence there would be a chance of continues breaking of the circuit.

So, AC relays are constructed with special mechanism such that continues magnetism is provided in order to avoid above problem. Such mechanisms include electronic circuit arrangement or shaded coil mechanism.

Working



- Relay works on the principle of electromagnetic induction.
- When the electromagnet is applied with some current it induces a magnetic field around it.
- Above image shows working of the relay .A switch is used to apply DC current to the load.
- In the relay Copper coil and the iron core acts as electromagnet.
- When the coil is applied with DC current it starts attracting the contact as shown. This is called energizing of relay.
- When the supply is removed it retrieves back to the original position. This is called De energizing of relay.

There are also such relays, whose contacts are initially closed and opened when there is supply i.e. exactly to opposite to the above shown relay.

Solid state relays will have sensing element to sense the input voltage and switches the output using opto-coupling.

Relay Contact Types

As we have seen that relay is a switch. The terminology “Poles and throws” is also applicable for relay. Depending on the number of contacts and number of circuits it switches relays can be classified.

Before we know about this classification of contacts we have to know the poles and throws of a relay switch.

Poles and Throws

Relays can switch one or more circuits. Each switch in relay is referred as pole. Number of circuits a relay connects is indicated by throws.

Depending on the poles and throws, relays are classified into

- Single pole single throw
- Single pole double throw
- Double pole single throw
- Double pole double throw

Single Pole Single Throw

A single pole single throw relay can control one circuit and can be connected to one output. It is used for the applications which require only ON or OFF state.

Single Pole Double Throw

A single pole double throw relay connects one input circuit to one of the two outputs. This relay is also called as changeover relay.

Though the SPDT has two output positions, it may consist of more than two throws depends on the configuration and requirement of the application.

Double pole single throw

A double pole single throw relay has two poles and single throw and it can be used to connect two terminals of a single circuit at a time. For example, this relay is used for connecting both phase and neutral terminals to the load at a time.

Double pole double throw

A DPDT (double pole double throw) relay has two poles and two throws for each pole. In motor direction control, these are used for phase or polarity reversal.

The switching action between contacts for all these relays is performed when the coil get energized as shown in figure below.

Relays can be classified into different types depending on their functionality, structure, application etc. Learn about different types of relays. Classification of Relays.

Relay Applications

Relays are used to protect the electrical system and to minimize the damage to the equipment connected in the system due to over currents/voltages. The relay is used for the purpose of protection of the equipment connected with it.

These are used to control the high voltage circuit with low voltage signal in applications audio amplifiers and some types of modems.

These are used to control a high current circuit by a low current signal in the applications like starter solenoid in automobile. These can detect and isolate the faults that occurred in power transmission and distribution system. Typical application areas of the relays include

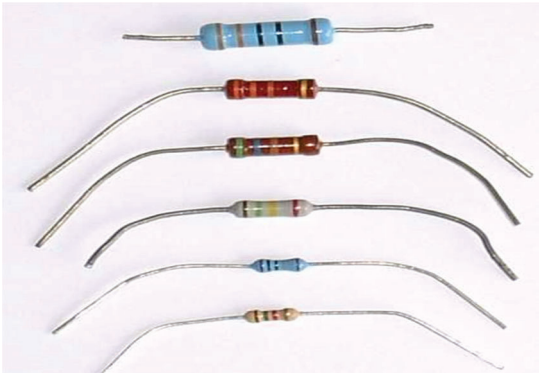
- Lighting control systems
- Telecommunication
- Industrial process controllers
- Traffic control
- Motor drives control
- Protection systems of electrical power system

Resistor

A resistor is a two-terminal electronic component designed to oppose an electric current by producing a voltage drop between its terminals in proportion to the current, that is, in accordance with Ohm's law:

$$V = IR$$

Resistors are used as part of electrical networks and electronic circuits. They are extremely commonplace in most electronic equipment. Practical resistors can be made of various compounds and films, as well as resistance wire (wire made of a high-resistivity alloy, such as nickel/chrome).



The primary characteristics of resistors are their resistance and the power they can dissipate. Other characteristics include temperature coefficient, noise, and inductance. Less well-known is critical resistance, the value below which power dissipation limits the maximum permitted current flow, and above which the limit is applied voltage. Critical resistance depends upon the materials constituting the resistor as well as its physical dimensions; it's determined by design.

Resistors can be integrated into hybrid and printed circuits, as well as integrated circuits. Size, and position of leads (or terminals) are relevant to equipment designers; resistors must be physically large enough not to overheat when dissipating their power.

A resistor is a two-terminal passive electronic component which implements electrical resistance as a circuit element. When a voltage V is applied across the terminals of a resistor, a current I will flow through the resistor in direct proportion to that voltage. The reciprocal of the constant of proportionality is known as the resistance R, since, with a given voltage V, a larger value of R further "resists" the flow of current I as given by Ohm's law:

$$I = \frac{V}{R}$$

Resistors are common elements of electrical networks and electronic circuits and are ubiquitous in most electronic equipment. Practical resistors can be made of various compounds and films, as well as resistance wire (wire made of a high-resistivity alloy, such as nickel-chrome). Resistors are also implemented within integrated circuits, particularly analog devices, and can also be integrated into hybrid and printed circuits.

The electrical functionality of a resistor is specified by its resistance: common commercial resistors are manufactured over a range of more than 9 orders of magnitude. When specifying that resistance in an electronic design, the required precision of the resistance may require attention to the manufacturing tolerance of the chosen resistor, according to its specific application. The temperature coefficient of the resistance may also be of concern in some precision applications. Practical resistors are also specified as having a maximum power rating which must exceed the anticipated power dissipation of that resistor in a particular circuit: this is mainly of concern in power electronics applications. Resistors with higher power ratings are physically larger and may require heat sinking. In a high voltage circuit, attention must sometimes be paid to the rated maximum working voltage of the resistor.

The series inductance of a practical resistor causes its behaviour to depart from ohms law; this specification can be important in some high-frequency applications for smaller values of resistance. In a low-noise amplifier or pre-amp the noise characteristics of a resistor may be an issue. The unwanted inductance, excess noise, and temperature coefficient are mainly dependent on the technology used in manufacturing the resistor. They are not normally specified individually for a particular family of resistors manufactured using a particular technology.[1] A family of discrete resistors is also characterized according to its form factor, that is, the size of the device and position of its leads (or terminals) which is relevant in the practical manufacturing of circuits using them.

Units

The ohm (symbol: Ω) is the SI unit of electrical resistance, named after Georg Simon Ohm. An ohm is equivalent to a volt per ampere. Since resistors are specified and manufactured over a very large range of values, the derived units of milliohm (1 mΩ = 10^{-3} Ω), kilohms (1 kΩ = 10^{3} Ω), and megohm (1 MΩ = 10^{6} Ω) are also in common usage.

The reciprocal of resistance R is called conductance G = 1/R and is measured in Siemens (SI unit), sometimes referred to as a mho. Thus a Siemens is the reciprocal of an ohm: $S = \Omega^{-1}$. Although the concept of conductance is often used in circuit analysis, practical resistors are always specified in terms of their resistance (ohms) rather than conductance.

standard values in series and/or parallel. This can also be used to obtain a resistance with a higher power rating than that of the individual resistors used. In the special case of N identical resistors all connected in series or all connected in parallel, the power rating of the individual resistors is thereby multiplied by N.

Power dissipation

The power P dissipated by a resistor (or the equivalent resistance of a resistor network) is calculated as:

$$P = I^2R = IV = \frac{V^2}{R}$$

The first form is a restatement of Joule's first law. Using Ohm's law, the two other forms can be derived.

The total amount of heat energy released over a period of time can be determined from the integral of the power over that period of time:

$$W = \int_{t_1}^{t_2} v(t)i(t)\,dt.$$

Practical resistors are rated according to their maximum power dissipation. The vast majority of resistors used in electronic circuits absorb much less than a watt of electrical power and require no attention to their power rating. Such resistors in their discrete form, including most of the packages detailed below, are typically rated as 1/10, 1/8, or 1/4 watt.

Resistors required to dissipate substantial amounts of power, particularly used in power supplies, power conversion circuits, and power amplifiers, are generally referred to as power resistors; this designation is loosely applied to resistors with power ratings of 1 watt or greater. Power resistors are physically larger and tend not to use the preferred values, colour codes, and external packages described below.

If the average power dissipated by a resistor is more than its power rating, damage to the resistor may occur, permanently altering its resistance; this is distinct from the reversible change in resistance due to its temperature coefficient when it warms. Excessive power dissipation may raise the temperature of the resistor to a point where it can burn the circuit board or adjacent components, or even cause a fire. There are flameproof resistors that fail (open circuit) before they overheat dangerously.

Note that the nominal power rating of a resistor is not the same as the power that it can safely dissipate in practical use. Air circulation and proximity to a circuit board, ambient temperature, and other factors can reduce acceptable dissipation significantly. Rated power dissipation may be given for an ambient temperature of 25 °C in free air. Inside an equipment case at 60 °C, rated dissipation will be significantly less; a resistor dissipating a bit less than the maximum figure given by the manufacturer may still be outside the safe operating areaand may prematurely fail.

Theory of operation

Ohm's law

The behaviour of an ideal resistor is dictated by the relationship specified in Ohm's law:

$$V = I \cdot R$$

Ohm's law states that the voltage (V) across a resistor is proportional to the current (I) passing through it, where the constant of proportionality is the resistance (R).

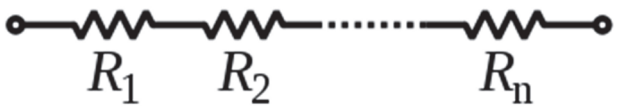
Equivalently, Ohm's law can be stated:

$$I = \frac{V}{R}$$

This formulation of Ohm's law states that, when a voltage (V) is present across a resistance (R), a current (I) will flow through the resistance. This is directly used in practical computations. For example, if a 300 ohm resistor is attached across the terminals of a 12 volt battery, then a current of 12 / 300 = 0.04 amperes (or 40 mill amperes) will flow through that resistor.

Series and parallel resistors

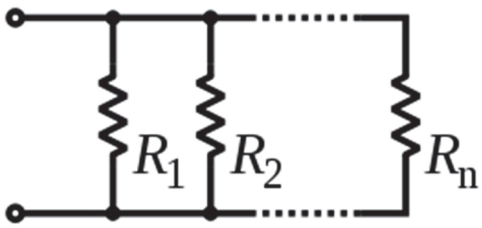
In a series configuration, the current through all of the resistors is the same, but the voltage across each resistor will be in proportion to its resistance. The potential difference (voltage) seen across the network is the sum of those voltages, thus the total resistance can be found as the sum of those resistances:



$$R_{\text{eq}} = R_1 + R_2 + \cdots + R_n$$

As a special case, the resistance of N resistors connected in series, each of the same resistance R, is given by NR.

Resistors in a parallel configuration are each subject to the same potential difference (voltage), however the currents through them add. The conductance of the resistors then add to determine the conductance of the network. Thus the equivalent resistance (R_{eq}) of the network can be computed:



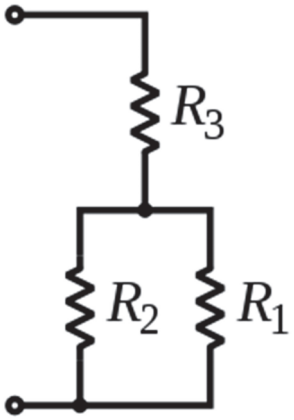
$$\frac{1}{R_{\text{eq}}} = \frac{1}{R_1} + \frac{1}{R_2} + \cdots + \frac{1}{R_n}$$

The parallel equivalent resistance can be represented in equations by two vertical lines "||" (as in geometry) as a simplified notation. For the case of two resistors in parallel, this can be calculated using:

$$R_{\text{eq}} = R_1 \| R_2 = \frac{R_1 R_2}{R_1 + R_2}$$

As a special case, the resistance of N resistors connected in parallel, each of the same resistance R, is given by R/N.

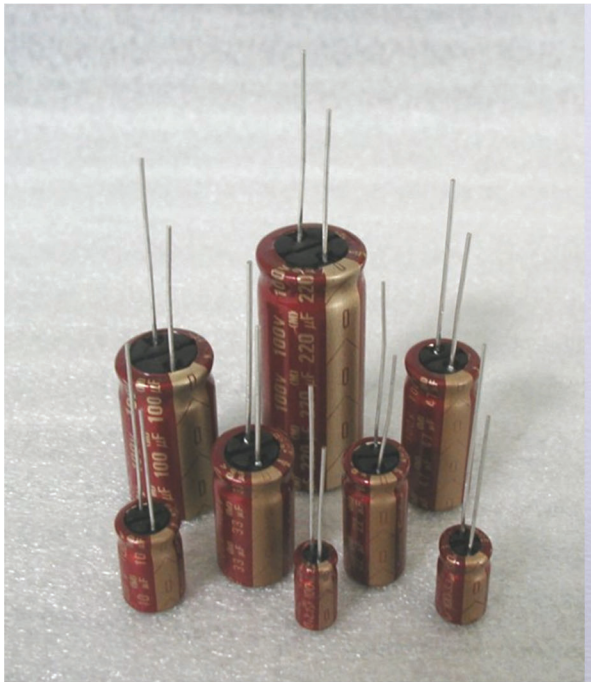
A resistor network that is a combination of parallel and series connections can be broken up into smaller parts that are either one or the other. For instance,



$$R_{\text{eq}} = (R_1 \| R_2) + R_3 = \frac{R_1 R_2}{R_1 + R_2} + R_3$$

Capacitor

A capacitor or condenser is a passive electronic component consisting of a pair of conductors separated by a dielectric. When a voltage potential difference exists between the conductors, an electric field is present in the dielectric. This field stores energy and produces a mechanical force between the plates. The effect is greatest between wide, flat, parallel, narrowly separated conductors.



An ideal capacitor is characterized by a single constant value, capacitance, which is measured in farads. This is the ratio of the electric charge on each conductor to the potential difference between them. In practice, the dielectric between the plates passes a small amount of leakage current. The conductors and leads introduce an equivalent series resistance and the dielectric has an electric field strength limit resulting in a breakdown voltage.

The properties of capacitors in a circuit may determine the resonant frequency and quality factor of a resonant circuit, power dissipation and operating frequency in a digital logic circuit, energy capacity in a high-power system, and many other important aspects.

A capacitor (formerly known as condenser) is a device for storing electric charge. The forms of practical capacitors vary widely, but all contain at least two conductors separated by a non-conductor. Capacitors used as parts of electrical systems, for example, consist of metal foils separated by a layer of insulating film.

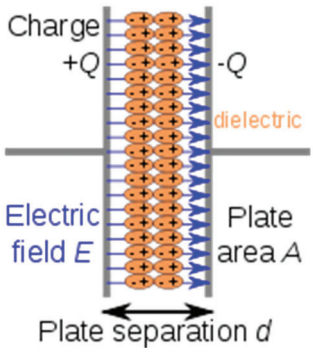
Capacitors are widely used in electronic circuits for blocking direct current while allowing alternating current to pass, in filter networks, for smoothing the output of power supplies, in the resonant circuits that tune radios to particular frequencies and for many other purposes.

A capacitor is a passive electronic component consisting of a pair of conductors separated by a dielectric (insulator). When there is a potential difference (voltage) across the conductors, a static electric field develops in the dielectric that stores energy and produces a mechanical force between the conductors. An ideal capacitor is characterized by a single constant value, capacitance, measured in farads. This is the ratio of the electric charge on each conductor to the potential difference between them.

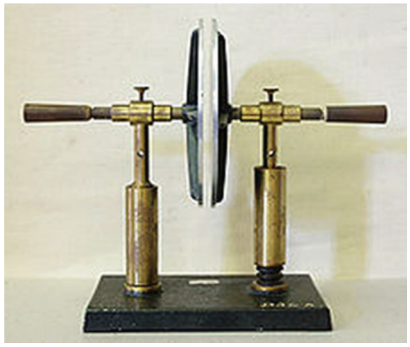
The capacitance is greatest when there is a narrow separation between large areas of conductor; hence capacitor conductors are often called "plates", referring to an early means of construction. In practice the dielectric between the plates passes a small amount of leakage current and also has an electric field strength limit, resulting in a breakdown voltage, while the conductors and leads introduce an undesired inductance and resistance.

Theory of operation

Capacitance



Charge separation in a parallel-plate capacitor causes an internal electric field. A dielectric (orange) reduces the field and increases the capacitance.



(A simple demonstration of a parallel-plate capacitor)

A capacitor consists of two conductors separated by a non-conductive region The non-conductive region is called the dielectric or sometimes the dielectric medium. In simpler terms, the dielectric is just an electrical insulator. Examples of dielectric mediums are glass, air, paper, vacuum, and even a semiconductordepletion region chemically identical to the conductors. A capacitor is assumed to be self-contained and isolated, with no net electric charge and no influence from any external electric field. The conductors thus hold equal and opposite charges on their facing surfaces, and the dielectric develops an electric field. In SI units, a capacitance of one farad means that one coulomb of charge on each conductor causes a voltage of one volt across the device. The capacitor is a reasonably general model for electric fields within electric circuits. An ideal capacitor is wholly characterized by a constant capacitance C, defined as the ratio of charge ±Q on each conductor to the voltage V between them:

$$C = \frac{Q}{V}$$

Sometimes charge build-up affects the capacitor mechanically, causing its capacitance to vary. In this case, capacitance is defined in terms of incremental changes:

$$C = \frac{\mathrm{d}q}{\mathrm{d}v}$$

Energy storage

Work must be done by an external influence to "move" charge between the conductors in a capacitor. When the external influence is removed the charge separation persists in the electric field and energy is stored to be released when the charge is allowed to return to its equilibrium position. The work done in establishing the electric field, and hence the amount of energy stored, is given by:

$$W = \int_{q=0}^{Q} V\,\mathrm{d}q = \int_{q=0}^{Q} \frac{q}{C}\mathrm{d}q = \frac{1}{2}\frac{Q^2}{C} = \frac{1}{2}CV^2 = \frac{1}{2}VQ.$$

Current-voltage relation

The current i(t) through any component in an electric circuit is defined as the rate of flow of a charge q(t) passing through it, but actual charges, electrons, cannot pass through the dielectric layer of a capacitor, rather an electron accumulates on the negative plate for each one that leaves the positive plate, resulting in an electron depletion and consequent positive charge on one electrode that is equal and opposite to the accumulated negative charge on the other. Thus the charge on the electrodes is equal to the integral of the current as well as proportional to the voltage as discussed above. As with any ant derivative, a constant of integration is added to represent the initial voltage v (t_0). This is the integral form of the capacitor equation,

$$v(t) = \frac{q(t)}{C} = \frac{1}{C}\int_{t_0}^{t} i(\tau)\mathrm{d}\tau + v(t_0)$$

.

Taking the derivative of this, and multiplying by C, yields the derivative form,

$$i(t) = \frac{\mathrm{d}q(t)}{\mathrm{d}t} = C\frac{\mathrm{d}v(t)}{\mathrm{d}t}$$

.

The dual of the capacitor is the inductor, which stores energy in the magnetic field rather than the electric field. Its current-voltage relation is obtained by exchanging current and voltage in the capacitor equations and replacing C with the inductance L.

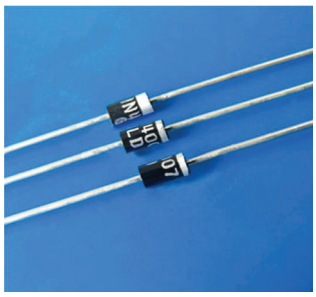
Diode

Diodes are used to convert AC into DC these are used as half wave rectifier or full wave rectifier. Three points must he kept in mind while using any type of diode.

1.Maximum forward current capacity

2.Maximum reverse voltage capacity

3.Maximum forward voltage capacity



(Fig: 1N4007 diodes)

The number and voltage capacity of some of the important diodes available in the market are as follows:

- Diodes of number IN4001, IN4002, IN4003, IN4004, IN4005, IN4006 and IN4007 have maximum reverse bias voltage capacity of 50V and maximum forward current capacity of 1 Amp.
- Diode of same capacities can be used in place of one another. Besides this diode of more capacity can be used in place of diode of low capacity but diode of low capacity cannot be used in place of diode of high capacity. For example, in place of IN4002; IN4001 or IN4007 can be used but IN4001 or IN4002 cannot be used in place of IN4007.The diode BY125made by company BEL is equivalent of diode from IN4001 to IN4003. BY 126 is equivalent to diodes IN4004 to 4006 and BY 127 is equivalent to diode IN4007.

PN JUNCTION OPERATION

Now that you are familiar with P- and N-type materials, how these materials are joined together to form a diode, and the function of the diode, let us continue our discussion with the operation of the PN junction. But before we can understand how the PN junction works, we must first consider current flow in the materials that make up the junction and what happens initially within the junction when these two materials are joined together.

Current Flow in the N-Type Material

Conduction in the N-type semiconductor, or crystal, is similar to conduction in a copper wire. Thatis, with voltage applied across the material, electrons will move through the crystal just as current wouldflow in a copper wire. This is shown in figure 1-15. The positive potential of the battery will attract the free electrons in the crystal. These electrons will leave the crystal and flow into the positive terminal of the battery. As an electron leaves the crystal, an electron from the negative terminal of the battery will enter the crystal, thus completing the current path. Therefore, the majority current carriers in the N-type material (electrons) are repelled by the negative side of the battery and move through the crystal toward the positive side of the battery.

Current Flow in the P-Type Material

Current flow through the P-type material is illustrated. Conduction in the P material is by positive holes, instead of negative electrons. A hole moves from the positive terminal of the P material to the negative terminal. Electrons from the external circuit enter the negative terminal of the material and fill holes in the vicinity of this terminal. At the positive terminal, electrons are removed from the covalent bonds, thus creating new holes. This process continues as the steady stream of holes (hole current) moves toward the negative terminal

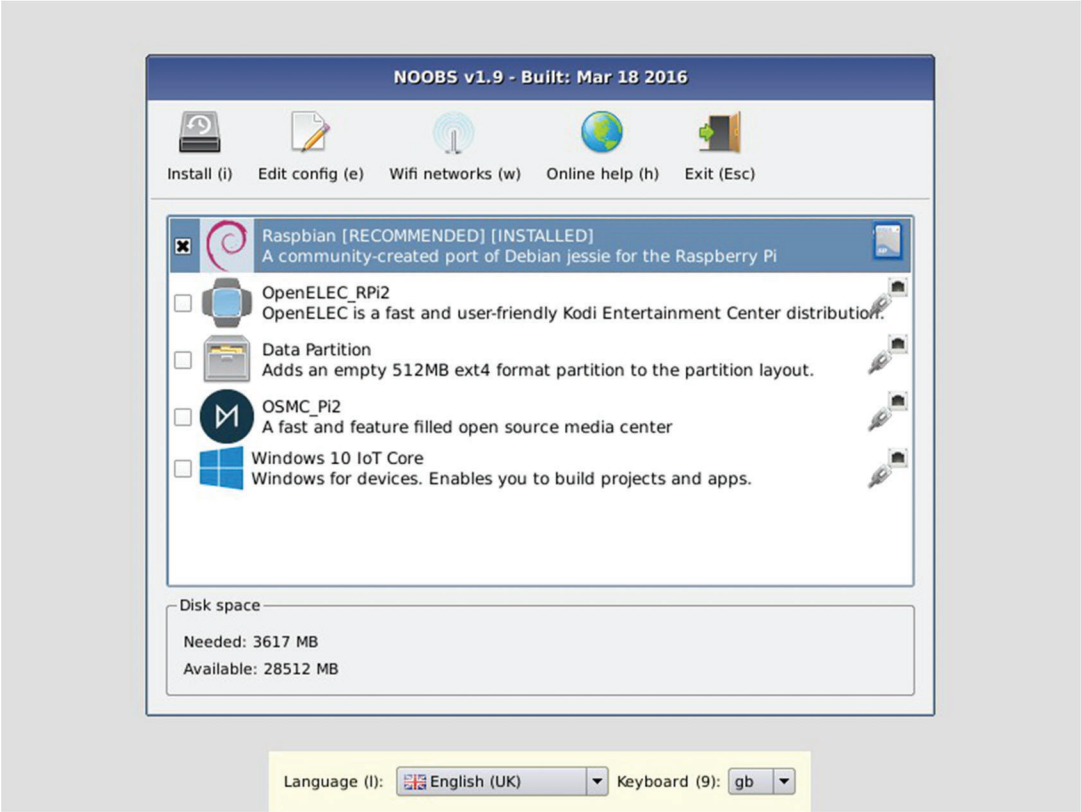
Software Description and Setup

NOOBS

NOOBS is designed to make it easy to select and install operating systems for the Raspberry Pi without having to worry about manually imaging your SD card.

The latest official release of NOOBS can be downloaded from http://downloads.raspberrypi.org/NOOBS_latest

For information on previous releases and version change lists,
visit https://github.com/raspberrypi/noobs/releases



About

On first boot NOOBS will format your SD card and allow you to select which OSes you want to install from a list. This OS list is automatically generated from both locally available OSes (i.e. those contained in the /os directory on disk) or those available from our remote repository (network connection required). Only the latest version of each OS will ever be displayed meaning that you can be sure that you have installed the most up-to-date release of your selected OS.

On any subsequent boot you can then press the SHIFT key to enter the NOOBS interface and easily reinstall your choice of OS.

The NOOBS interface provides the following functionality:

- **Install**: Installs the selected OS onto your SD card. Changing this selection erases all OSes currently installed.
- **Edit Config**: Opens a text editor allowing the cmd line and config for the selected installed OS to be edited.
- **Online Help**: [Networking Required] Open a browser that displays the Raspberry Pi Help page (http://www.raspberrypi.org/help/), allowing people to quickly access help and troubleshooting.
- **Exit**: Quits NOOBS and reboots the Pi into the OS boot menu.
- **Language Selection**: Allows you to select the language to be displayed.
- **Keyboard Layout Selection**: Allows you to select the keyboard layout to be used.
- **Display Mode Selection**: By default, NOOBS will output over HDMI at your display's preferred resolution, even if no HDMI display is connected. If you do not see any output on your HDMI display or are using the composite output, press 1, 2, 3 or 4 on your keyboard to select HDMI preferred mode, HDMI safe mode, composite PAL mode or composite NTSC mode respectively.

Note that all user settings (language, keyboard layout, display mode) will persist between reboots and will also be automatically passed to the installed OS. This means that if you can see the NOOBS interface on your display device then you should be able to see the OS CLI/GUI when it boots too!

Setup

To setup a blank SD card with NOOBS:

- Format an SD card that is 4GB or greater in size as FAT (see instructions on how to do this below)
- Download and extract the files from the NOOBS zip file. (Windows built-in zip features may have trouble with this file. If so, use another program such as 7zip.)
- Copy the extracted files onto the SD card that you just formatted so that this file is at the root directory of the SD card. **Please note that in some cases it may extract the files into a folder, if this is the case then please copy across the files from inside the folder rather than the folder itself.**

On first boot the "RECOVERY" FAT partition will be automatically resized to a minimum and a list of OSes that are available to install will be displayed.

Operating System Choice

NOOBS is available in 2 formats:

- NOOBS Full includes the installation files for Raspbian only.
- NOOBS-Lite does not include any Operating Systems at all.

OS Network Download

Both versions of NOOBS allow additional Operating Systems to be downloaded from our remote repository. To do this, the Raspberry Pi must be connected to a wired network, or it can connect over Wifi using the Raspberry Pi USB wifi dongle or the Raspberry Pi 3 Model B built-in wifi.

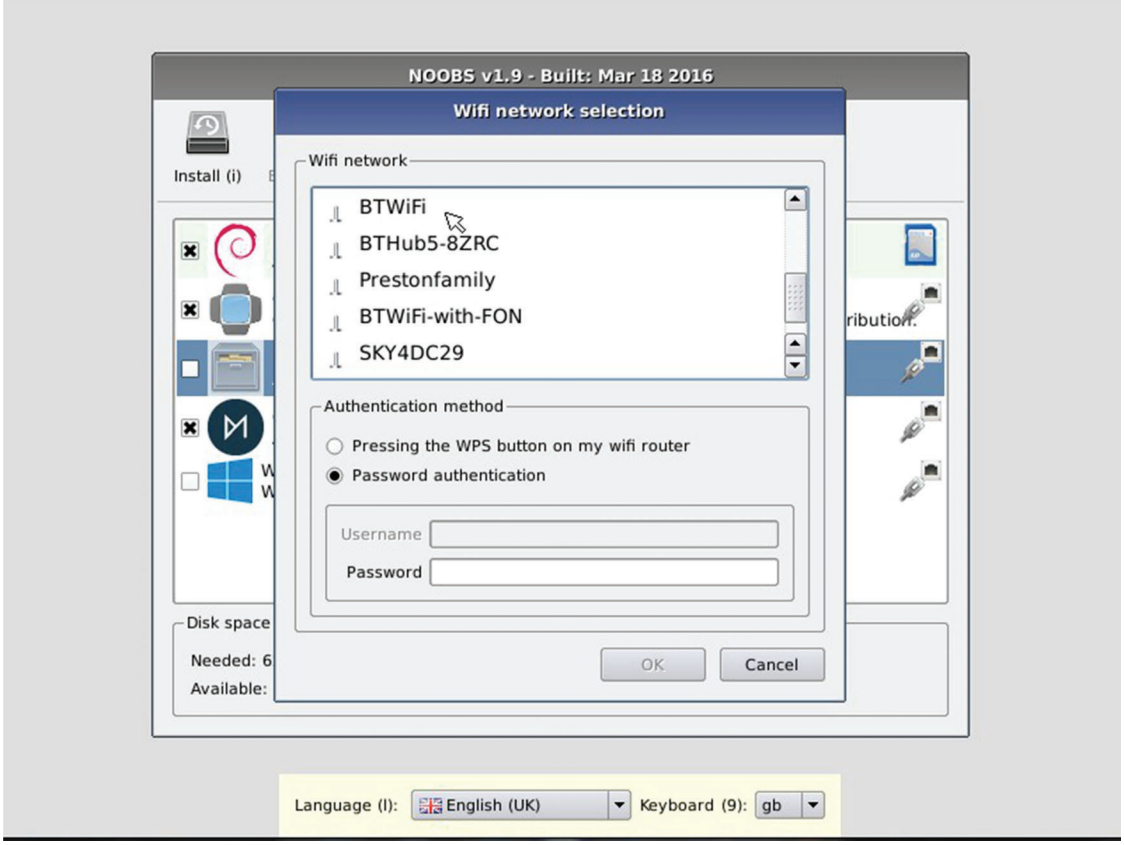
Once connected, the Pi will only show a list of Operating Systems that are appropriate to your Pi Model. If you want to see ALL available OS, edit the recovery.cmdline file in the root NOOBS directory and append showall to the arguments list.

Wired Networks

If a wired ethernet cable is plugged into the Pi before NOOBS starts, NOOBS will connect via DHCP to our remote download repository and present a list of available Operating Systems that are available for installation.

Wifi Networks

If you have the official Rapberry Pi USB wifi Dongle, or are using the Raspberry Pi 3 Model B with built-in wifi, the wifi icon on the NOOBS toolbar will be available. Click on this to select your Wifi SSID network and enter the wifi password.



How to Format an SD card as FAT

For **Windows** users, we recommend formatting your SD card using the SD Association's Formatting Tool, which can be downloaded from https://www.sdcard.org/downloads/formatter_4/ You will need to set "FORMAT SIZE ADJUSTMENT" option to "ON" in the "Options" menu to ensure that the entire SD card volume is formatted - not just a single partition. For more detailed and beginner-friendly formatting instructions, please refer to http://www.raspberrypi.org/quick-start-guide

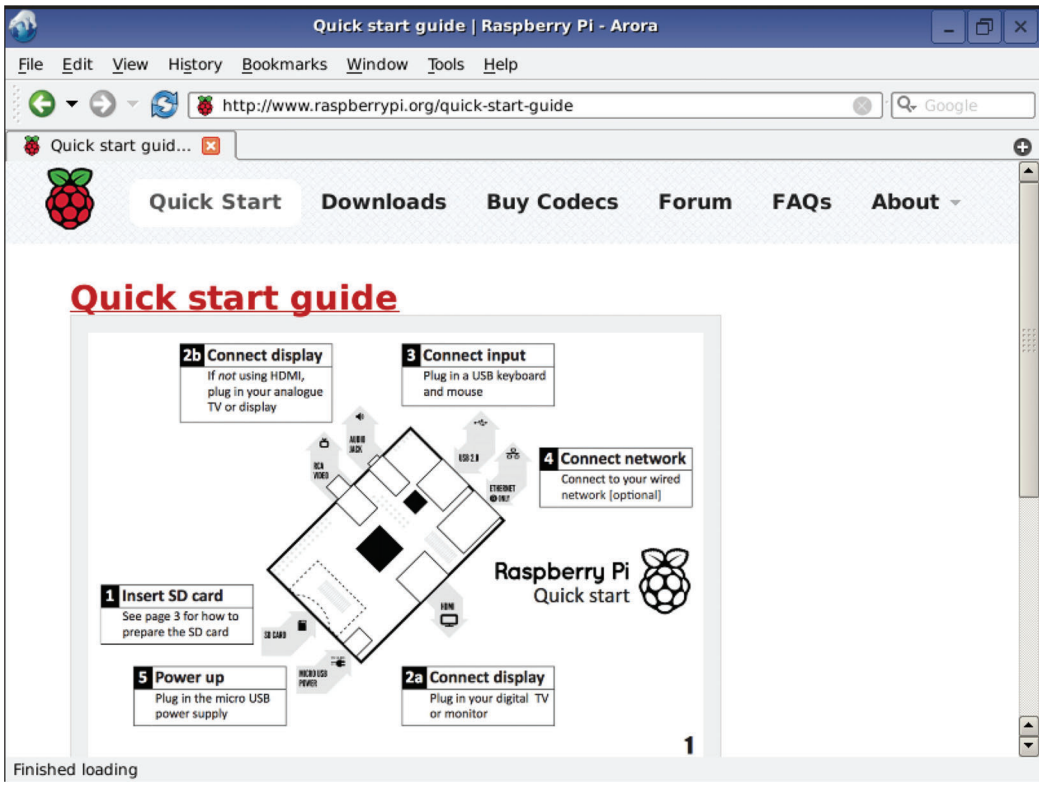
The SD Association's Formatting Tool is also available for **Mac** users although the default OSX Disk Utility is also capable of formatting the entire disk (select the SD card volume and choose "Erase" with "MS-DOS" format).

For **Linux** users we recommend gparted (or the command line version parted). (Update: Norman Dunbar has written up the following formatting instructions for Linux users: http://qdosmsq.dunbar-it.co.uk/blog/2013/06/noobs-for-raspberry-pi/)

Screenshots

OS Installation

Simply select the checkbox next to each OS you want to install using either a mouse or keyboard (arrow keys to traverse the list, enter to toggle the selected OS's checkbox), then click the "Install" icon (or press "i" on your keyboard) to install the selection. The icons shown on the right of the list indicate whether the OS is being installed from the SD card (SD card icon) or from the online OS repository (Ethernet icon).



Easy Config File Editor

The built-in config file editor allows you to edit the config file of the OS currently highlighted in the OS list. This allows you to easily add license keys to different OS installs through the same interface.

Note that the output mode selected by the user through pressing one of number keys 1 to 4 (for HDMI preferred, HDMI VGA, Composite PAL and Composite NTSC respectively), will be automatically set in the config.txt files of your installed OSes. This means that you shouldn't have to worry about manually changing your display settings to get your installed OS to display correctly on your display device.

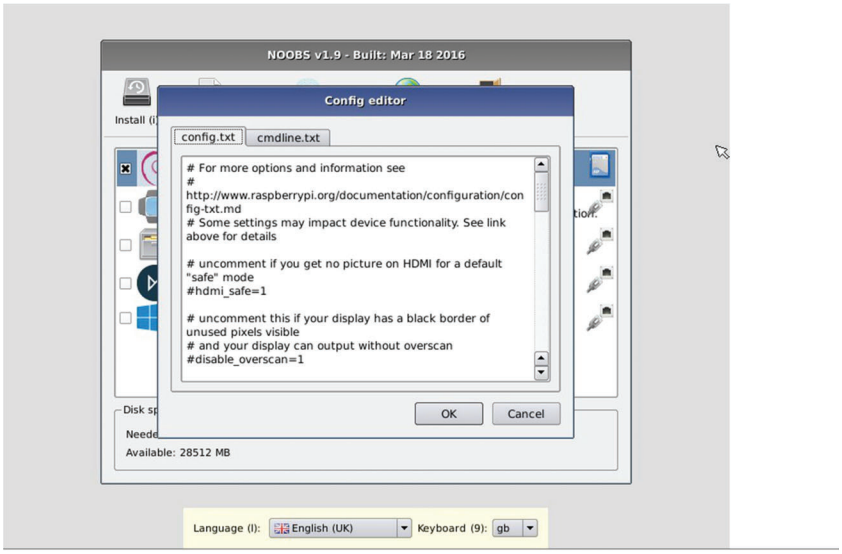
Installer Slideshow

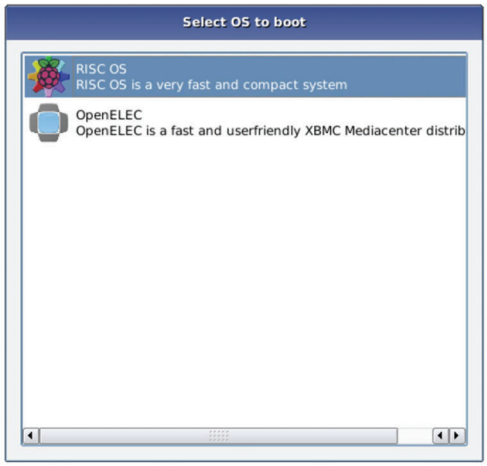
An installer slideshow guides you through your first steps with each OS while it installs.

OS Boot Selector

After multiple OSes have been installed, you can select which OS to boot through this selection window that is automatically displayed. NOOBS will remember your choice and boot this OS by default unless a different option has been selected within 10 seconds.

Note that if only one OS is installed then the boot selector will not be displayed and the OS will be automatically booted.





Advanced Usage (for experts and teachers)

How to Automatically Install an OS

Even if you are using your Pi without a display, you can still use NOOBS to easily install an OS of your choice. To set up NOOBS to automatically and silently (i.e. without requiring any user input) install a specific OS, follow these steps:

1. Copy the OS folder for the OS you want to install into the /os dir (or alternatively delete all other OSes contained in the/os dir so that only your chosen OS remains.
2. If the OS you want to automatically install has multiple flavours available, edit the flavours.json file so that it only contains the flavour entry that you want to install.
3. Edit the recovery.cmdline file in the root NOOBS directory and append silent install to the arguments list.

When you now boot your Pi using an SD card containing the modified version of NOOBS that you just created, it will automatically install the OS you chose and boot into it after the installation has finished.

Preconfiguring a WiFi network

If you already know your WiFi details, you can preconfigure NOOBS to use it straight away. Put a copy of your wpa_supplicant.conf file on the NOOBS root partition and NOOBS will read it and store it in its settings for all future uses.

How to create a custom OS version

There are two main use cases for which you may want to create a custom version of one of the standard OS releases that is suitable for installation via NOOBS:

- If you are a teacher wanting to easily deploy a custom OS release containing pre-defined set of packages and files onto a number of SD cards (e.g. to provision a class set of Raspberry Pi's or quickly restore a Raspberry Pi back to custom "factory" settings).
- If you want to be able to back up your existing installed packages and files so that any future OS re-install does not force you back to a clean install.

The following steps allow you to create a modified copy of one of the standard OS releases that contains your custom files, packages and settings.

1. Download a base version of NOOBS from http://downloads.raspberrypi.org/NOOBS_latest
2. Extract the NOOBS zip file
3. Navigate to the os directory
4. Create a copy of the folder containing the OS release that you want to modify and rename it with a custom name.
5. Edit the following fields in the os.json file contained in the folder that you just created
    i. "name" - replace the name of the base OS with the name of your custom OS version
    ii. "description" - replace the description of the standard OS install with one for your custom OS version
6. [Optional] Rename or replace the existing <OS>.png icon file with one matching the name of your custom OS version
7. [Optional] Replace the PNG image files in the slides and slides_vga directory with your own custom installer slides
8. Edit the following fields in the partitions.json file contained in the folder that you just created
    i. "partition_size_nominal" - replace the numerical value with the size of the paritions in your custom OS version
    ii. "uncompressed_tarball_size" - replace the numerical value with the size of your filesystem tarballs when uncompressed

9. Replace the .tar.xz root and boot filesystem tarballs with copies created from your custom OS version (these instructions assume you're only using a single OS at a time with NOOBS - they won't work if you're running multiple OSes from a single SD card). The name of these tarballs needs to match the labels given in partitions.json.
    i. To create the root tarball you will need to run tar -cvpf <label>.tar /* --exclude=proc/* --exclude=sys/* --exclude=dev/pts/* from within the root filesystem of your custom OS version. You should then compress the resulting tarball with xz -9 -e <label>.tar.
    ii. To create the boot tarball you will need to run tar -cvpf <label>.tar . at the root directory of the boot partition of your custom OS version. You should then compress the resulting tarball with xz -9 -e <label>.tar.

How to change the default Language, Keyboard layout, Display mode or Boot Partition etc.

Edit the recovery.cmdline file in the root NOOBS directory and append the following arguments where relevant:

- lang=<two-letter language code> (e.g. lang=de or lang=en)
- keyboard=<two-letter layout code> (e.g. keyboard=de or keyboard=us)
- display=<display mode number> (e.g. display=1 or display=3)
- partition=<partition_number> (e.g. partition=6)
- showall (shows all available OSes regardless of your Raspberry Pi model)

Note that these defaults will be overwritten by any changes made in the GUI to these settings.

How to bypass the Recovery splashscreen and boot directly into a fixed partition

After you have installed your chosen OSes, add the following file to the root directory of NOOBS to force the indicated partition to be booted at power-on.

1. Add a text file named autoboot.txt to the root directory of NOOBS.
2. Add boot_partition=<partition number> to the file and save it to disk.

This will also prevent the splashscreen from being displayed at boot. The partition number can be found by running sudo fdisk -l the partition will be one of the FAT32 partitions /dev/mmcblk0p6 would be partition 6. Note that once anautoboot.txt file is present, there's then no way to force the NOOBS GUI to display, until you delete (or rename) theautoboot.txt file.

How to use with the Raspberry Pi Touch Display

If NOOBS detects you are using the Raspberry Pi Touch Display, it will enable the following functionality:

- A tap on the touchscreen can activate NOOBS aswell as holding the shift key down.
- A tap on the touchscreen will simulate a mouse click
- A longpress on the touchscreen will simulate a mouse double-click.

Troubleshooting

What to do if your SHIFT keypress isn't detected

Try pressing shift only when the grey splashscreen is displayed rather than holding it from boot up.

How to boot into "Safe Mode"

To boot into a basic busybox shell rather than launching the NOOBS GUI, you can *either*:

1. Append rescueshell to the argument list in the recovery.cmdline file which is found in the root NOOBS directory.
2. Insert a physical jumper between pins 5 & 6 of GPIO header P1. If you have external hardware or an addon board connected to the GPIO header, you may find that pin 5 is being pulled low and accidentally triggering "Safe Mode". To prevent this you can append disablesafemode to the argument list in the recovery.cmdline file which is found in the root NOOBS directory.

How to enable using the GPIO to trigger entering Recovery Mode

To force Recovery Mode to be entered on boot and to show the NOOBS interface, you normally press the SHIFT key during bootup. If you don't have a keyboard or the SHIFT keypress isn't being detected, you should complete the following steps to force the NOOBS interface to be displayed on boot:

1. Append gpiotriggerenable to the argument list in the recovery.cmdline file which is found in the root NOOBS directory
2. Reboot

To force Recovery Mode being entered on boot, connect GPIO pin 3 on header P1 to GND (pin 25). If GPIO pin 3 remains unconnected then it will boot through to the installed OS as normal.

How to force Recovery Mode being entered on boot (overrides GPIO or keyboard input)

Alternatively, if you are unable to use either the GPIO or keyboard to trigger entering Recovery Mode, you can:

1. Append forcetrigger to the argument list in the recovery.cmdline file which is found in the root NOOBS directory.
2. Reboot

Note that with this option enabled, the Recovery Mode will be displayed **every** time you boot from your NOOBS card (until you edit recovery.cmdline again)

How to disable using the keyboard to trigger entering Recovery Mode

In some rare cases, you may find that NOOBS incorrectly detects a SHIFT keypress from your keyboard regardless of the presence of user input. In such cases it may be helpful to disable using the keyboard to trigger Recovery Mode being entered.
To prevent a SHIFT keypress from entering Recovery Mode on boot (maybe you have a problematic keyboard which is erroneously triggering every time you boot), you can:

1. Append keyboardtriggerdisable to the argument list in the recovery.cmdline file which is found in the root NOOBS directory.
2. Reboot

How to change display output modes

By default, NOOBS will output over HDMI at your display's preferred resolution, even if no HDMI display is connected. If you do not see any output on your HDMI display or are using the composite output, press 1, 2, 3 or 4 on your keyboard to select HDMI preferred mode, HDMI safe mode, composite PAL mode or composite NTSC mode respectively.

If you don't have a keyboard, you can still change the display mode used by NOOBS through editing the recovery.cmdlinefile in the root NOOBS directory prior to first boot and appending the following argument:

- display=<display mode number> (e.g. display=1 or display=3)

How to Rebuild NOOBS

Note that this will require a minimum of 6GB free disk space.

Get Build Dependencies

On Ubuntu:

sudo apt-get install build-essential rsync texinfo libncurses-dev whois unzip bc qt4-linguist-tools

Run Build Script

./BUILDME.sh

Buildroot will then build the software and all dependencies, putting the result in the output directory.

Buildroot by default compiles multiple files in parallel, depending on the number of CPU cores you have.

If your build machine does have a quad core CPU, but relatively little RAM, you may want to lower the number to prevent swapping:

- cd buildroot ; make menuconfig
- "Build options" -> "Number of jobs to run simultaneously"

If your build machine also has some QT5 components, it is useful to export QT_SELECT=4 before building to ensure the QT4 component versions are selected.

How to run your Build

In order to setup an SD card with a newly built version of NOOBS, you will need to:

- Format an SD card that is 4GB or greater in size as FAT
- Replace the /os directory in /output with the copy contained in the release version of NOOBS (see above for download links)
- Copy the files in the /output directory onto the SD card

About the Buildroot infrastructure

To add extra packages: cd buildroot ; make menuconfig

Recovery software packaging is in: buildroot/package/recovery

Kernel configuration used: buildroot/kernelconfig-recovery.armv6 and kernelconfig-recovery.armv7

Main differences with bcmrpi_defconfig:

- CONFIG_BLK_DEV_INITRD=y - initramfs support
- CONFIG_INPUT_EVDEV=y - evdev support built-in
- CONFIG_USB_HID=y - usb HID driver built-in
- All modules disabled.
- (This has changed significantly from v1.5 to use a squashfs)

Modifying Qt source

Source is in the recovery folder. Be aware that user interface screens will appear larger in Qt Creator then when deployed on the Pi, can raise font sizes 2 points to compensate.

Several constants can be changed in config.h

Wrap code that calls Qt Embedded specific classes (such as QWSServer) between

```
#ifdef Q_WS_QWS
```

and

```
#endif
```

so that the project also compiles and can be tested under standard Qt.

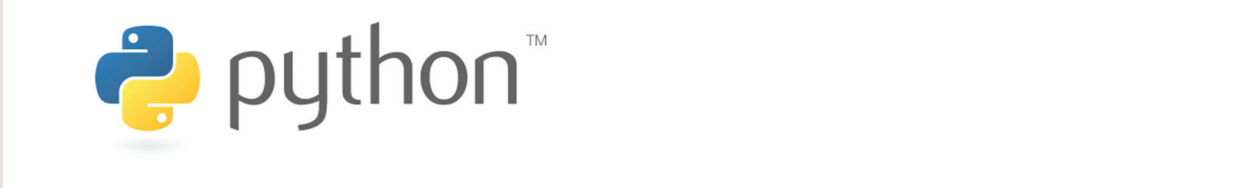
Python Compiler

IDLE is Python's Integrated Development and Learning Environment.

IDLE has the following features:

- coded in 100% pure Python, using the **tkinter** GUI toolkit
- cross-platform: works mostly the same on Windows, Unix, and Mac OS X
- Python shell window (interactive interpreter) with colorizing of code input, output, and error messages
- multi-window text editor with multiple undo, Python colorizing, smart indent, call tips, auto completion, and other features
- search within any window, replace within editor windows, and search through multiple files (grep)
- debugger with persistent breakpoints, stepping, and viewing of global and local namespaces
- configuration, browsers, and other dialogs

Python is a wonderful and powerful programming language that's easy to use (easy to read **and** write) and with Raspberry Pi lets you connect your project to the real world.



Python syntax is very clean, with an emphasis on readability and uses standard English keywords. Start by opening IDLE from the desktop.

IDLE

The easiest introduction to Python is through IDLE, a Python development environment. Open IDLE from the Desktop or applications menu:

IDLE gives you a REPL (Read-Evaluate-Print-Loop) which is a prompt you can enter Python commands in to. As it's a REPL you even get the output of commands printed to the screen without using print.

Note two versions of Python are available: Python 2 and Python 3. Python 3 is the newest version and is recommended, however Python 2 is available for legacy applications which do not support Python 3 yet. For the examples on this page you can use Python 2 or 3 (see Python 2 vs. Python 3).

You can use variables if you need to but you can even use it like a calculator. For example:

```
>>> 1 + 2
3
>>> name = "Sarah"
>>> "Hello " + name
'Hello Sarah'
```

IDLE also has syntax highlighting built in and some support for autocompletion. You can look back on the history of the commands you've entered in the REPL withAlt + P (previous) and Alt + N (next).

BASIC PYTHON USAGE

Hello world in Python:

```
print("Hello world")
```

Simple as that!

INDENTATION

Some languages use curly braces { and } to wrap around lines of code which belong together, and leave it to the writer to indent these lines to appear visually nested. However, Python does not use curly braces but instead requires indentation for nesting. For example a for loop in Python:

```
for i in range(10):
    print("Hello")
```

The indentation is necessary here. A second line indented would be a part of the loop, and a second line not indented would be outside of the loop. For example:

```
for i in range(2):
    print("A")
    print("B")
```

would print:

```
A
B
A
B
```

whereas the following:

```
for i in range(2):
    print("A")
print("B")
```

would print:

```
A
A
B
```

VARIABLES

To save a value to a variable, assign it like so:

```
name = "Bob"
age = 15
```

Note here I did not assign types to these variables, as types are inferred, and can be changed (it's dynamic).

```
age = 15
age += 1  # increment age by 1
print(age)
```

This time I used comments beside the increment command.

COMMENTS

Comments are ignored in the program but there for you to leave notes, and are denoted by the hash # symbol.
Multi-line comments use triple quotes like so:

```
"""
This is a very simple Python program that prints "Hello".
That's all it does.
"""

print("Hello")
```

LISTS

Python also has lists (called arrays in some languages) which are collections of data of any type:

```
numbers = [1, 2, 3]
```

Lists are denoted by the use of square brackets [] and each item is separated by a comma.

ITERATION

Some data types are iterable, which means you can loop over the values they contain. For example a list:

```
numbers = [1, 2, 3]

for number in numbers:
    print(number)
```

This takes each item in the list numbers and prints out the item:

```
1
2
3
```

Note I used the word number to denote each item. This is merely the word I chose for this - it's recommended you choose descriptive words for variables - using plurals for lists, and singular for each item makes sense. It makes it easier to understand when reading.

Other data types are iterable, for example the string:

```
dog_name = "BINGO"

for char in dog_name:
    print(char)
```

This loops over each character and prints them out:

```
B
I
N
G
O
```

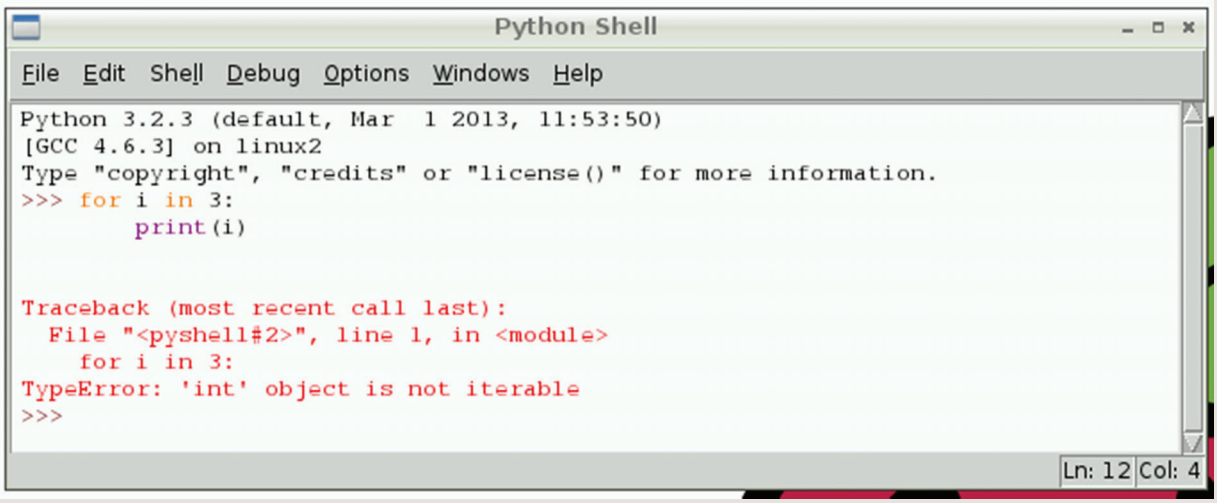
RANGE

The integer data type is not iterable and tryng to iterate over it will produce an error. For example:

```
for i in 3:
    print(i)
```

will produce:

TypeError: 'int' object is not iterable



However you can make an iterable object using the range function:

```
for i in range(3):
    print(i)
```

range(5) contains the numbers 0, 1, 2, 3 and 4 (five numbers in total). To get the numbers 1 to 5 use range(1, 6).

LENGTH

You can use functions like len to find the length of a string or a list:

```
name = "Jamie"
print(len(name))  # 5

names = ["Bob", "Jane", "James", "Alice"]
print(len(names))  # 4
```

IF STATEMENTS

You can use if statements for control flow:

```
name = "Joe"

if len(name) > 3:
    print("Nice name,")
    print(name)
else:
    print("That's a short name,")
    print(name)
```

PYTHON FILES IN IDLE

To create a Python file in IDLE, click File > New File and you'll be given a blank window. This is an empty file, not a Python prompt. You write a Python file in this window, save it, then run it and you'll see the output in the other window.

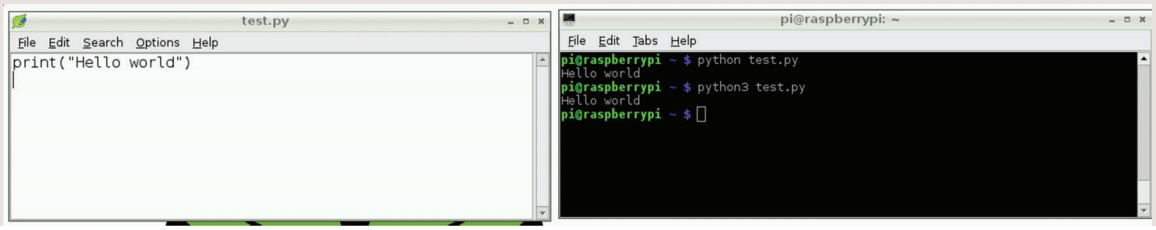
For example, in the new window, type:

```
n = 0

for i in range(1, 101):
    n += i

print("The sum of the numbers 1 to 100 is:")
print(n)
```

Then save this file (File > Save or Ctrl + S) and run (Run > Run Moduleor hit F5) and you'll see the output in your original Python window.

EXECUTING PYTHON FILES FROM THE COMMAND LINE

You can write a Python file in a standard editor like Vim, Nano or LeafPad, and run it as a Python script from the command line. Just navigate to the directory the file is saved (use cd and ls for guidance) and run with python, e.g.python hello.py.



MORE ON PYTHON

PYTHON 2 VS. PYTHON 3

The short version: Python 2 is legacy, Python 3 is the present and future of the language.

Python 2 was released in 2000, and Python 3 was released in 2008. Python 3 is recommended, but some libraries have not yet been ported to Python 3 which is why Python 2 is still present and widely used.

If you are familiar with Python 2 but not Python 3, here is a summary of the basic key differences:

- Print
  - In Python 2, `print` is a statement and did not require brackets, e.g. `print "Hello"` .
  - In Python 3, `print` is a function, so you pass in what you want to print as parameters, e.g. `print("Hello")` or `print("My age is", age)` .
  - Using brackets for `print` in Python 2 works fine, so it's common to see this used for compatibility. However printing multiple objects in the same `print` command works differently.
  - In Python 3 this prints each one, space separated, and in Python 2 the collection of items is printed as a tuple, e.g. `("My age is", 15)`
- Input / Raw input
  - In Python 2, the function `raw_input` takes input from the user.
  - In Python 3, the function is called `input` .

- Integer division
  - In Python 2 `/` is used for exact integer division, always returning an integer. This means it returns the value of the number of times one number divides in to another whole, and ignores the remainder, e.g. `1 / 2` returns `0`, `2 / 2` returns `1` and `3 / 2` returns `1`.
  - In Python 3, `/` does 'true' division, returning a floating point number (decimal), e.g. `1 / 2` returns `0.5`.
  - To get floating point division in Python 2, convert one or both numbers to a float first, e.g. `1.0 / 2` which returns `0.5`.
  - To get exact integer division in Python 3, use `//`, e.g. `1 // 2` returns `0`.

Python 2.7.6 was released in 2013. The 2.x branch will have no further major releases.

Read more on the differences on the Python wiki

CONVENTION

While indentation is essential in Python, there are other aspects which are syntactically correct but considered bad practise. These are given in a style guide called PEP 8 and include conventions such as always using spaces around operators i.e. `a = 1 + 2` over `a=1+2` and a maximum line length of 79 characters. It also suggests using four spaces per tab (this is configurable in your editor).

The philosophy of Python is summarised in PEP 20 (The Zen of Python) which encourages good Pythonic writing style. For example:

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Readability counts

OTHER WAYS OF USING PYTHON

COMMAND LINE

The standard built-in Python REPL is accessed by typing `python` in the Terminal. Type `python3` for Python 3.

This REPL is a prompt ready for Python commands to be entered. You can use this the same as IDLE, but it does not have syntax highlighting or autocompletion. You can look back on the history of the commands you've entered in the REPL by using the Up/Down keys. Use `Ctrl + D` to exit.

IPYTHON

An alternative command line Python prompt is IPython. IPython is an interactive Python shell with syntax highlighting, autocompletion, pretty printing, built-in documentation and more. IPython is not installed by default. Install with:

```
sudo apt-get install ipython
```

or for Python 3:

```
sudo apt-get install ipython3
```

Then run with `ipython` or `ipython3` from the command line. It works like the standard `python`, but with more features. Try typing `len?` and hitting `Enter`. You're shown information including the docstring for the `len` function:

Type: builtin_function_or_method

String Form:<built-in function len>

Namespace: Python builtin

Docstring:

len(object) -> integer

Return the number of items of a sequence or mapping.

Try the following dictionary comprehension:

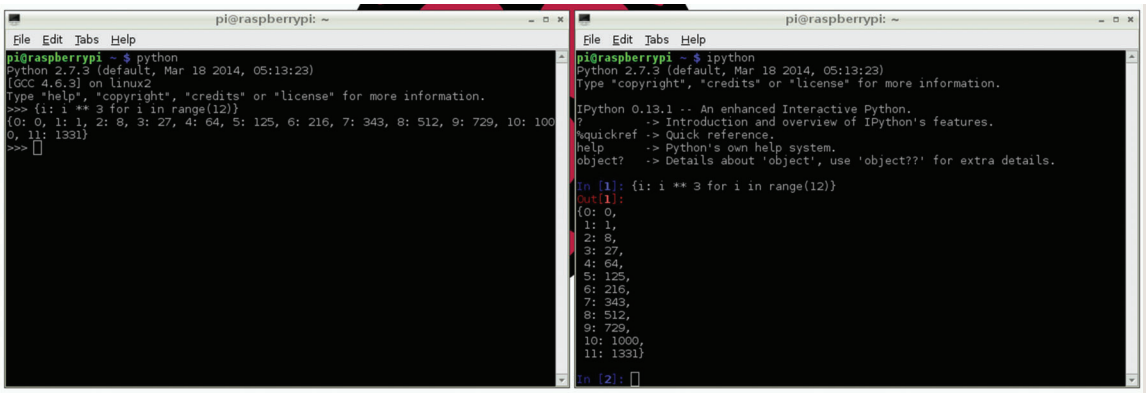
```
{i: i ** 3 for i in range(12)}
```

which will pretty print the following:

```
{1: 1,
 2: 8,
 3: 27,
 4: 64,
 5: 125,
 6: 216,
 7: 343,
 8: 512,
 9: 729,
 10: 1000,
 11: 1331}
```

In `python` this would have printed on one line:

{1: 1, 2: 8, 3: 27, 4: 64, 5: 125, 6: 216, 7: 343, 8: 512, 9: 729, 10: 1000, 11: 1331}



You can look back on the history of the commands you've entered in the REPL by using the Up/Down keys like in `python` but it also persists over sessions, so you can exit `ipython` and return (or switch between v2/3) and the history remains. Use `Ctrl + D` to exit.

INSTALLING PYTHON LIBRARIES

APT

Some Python packages can be found in the Raspbian archives, and can be installed using APT, for example:

```
sudo apt-get update
sudo apt-get install python-picamera
```

This is a preferable method of installing things as it means that the modules you install can be kept up to date easily with the usual `sudo apt-get update` and `sudo apt-get upgrade` commands.

PIP

Not all Python packages are available in the Raspbian archives, and those that are can sometimes be out of date. If you can't find a suitable version in the Raspbian archives you can install packages from the Python Package Index (also known as PyPI). To do so, use the `pip` tool (which is installed with the `python-pip` package in Raspbian:

```
sudo apt-get install python-pip
sudo pip install simplejson
```

Read more on installing software in Python

GPIO

Using Python on the Raspberry Pi opens up the opportunity to connect to the real world through the Pi's GPIO pins. This can be done with the RPi GPIO library. It is preinstalled on recent Raspbian images, but if you have an older one you can install it with:

```
sudo apt-get install python-rpi.gpio
```

or

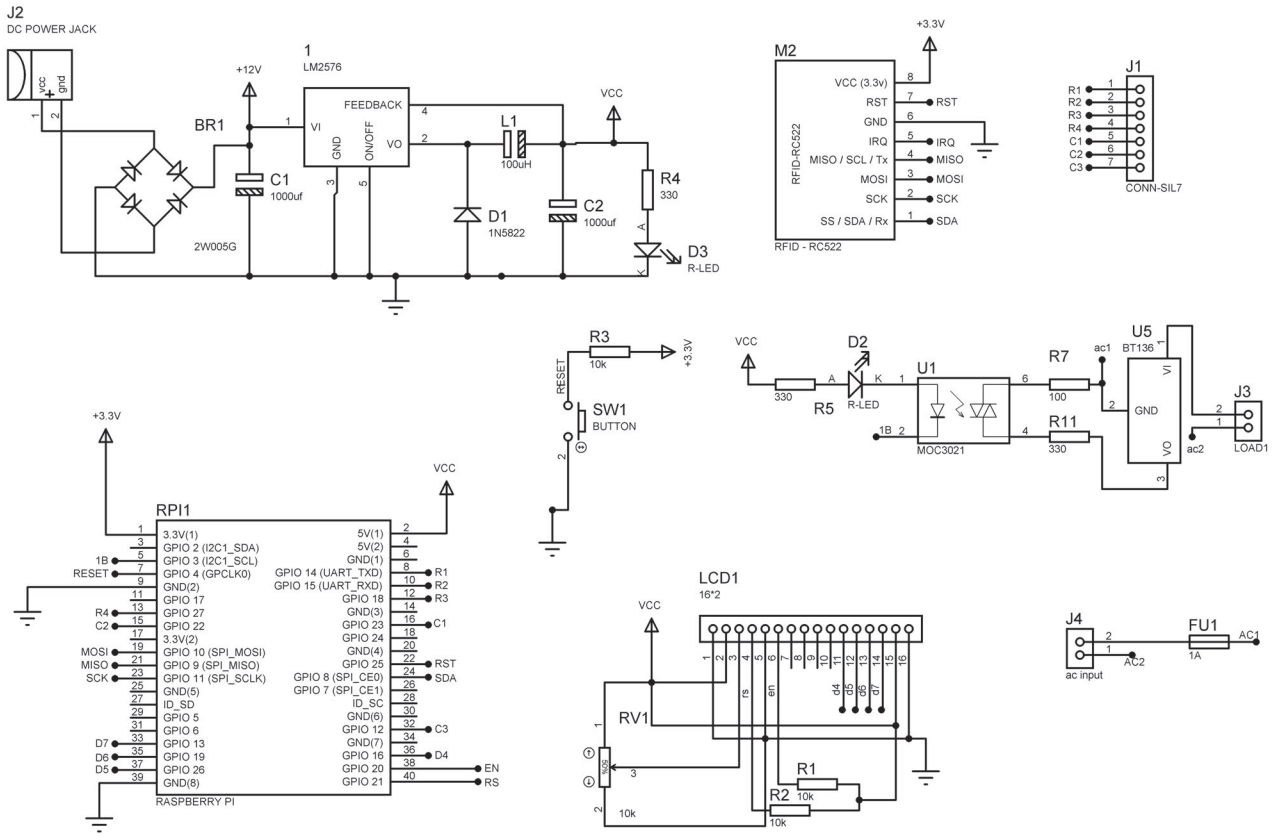
```
sudo apt-get install python3-rpi.gpio
```

To control the GPIO pins you'll need root access, so run `sudo python`, `sudo ipython` or `sudo idle &`.

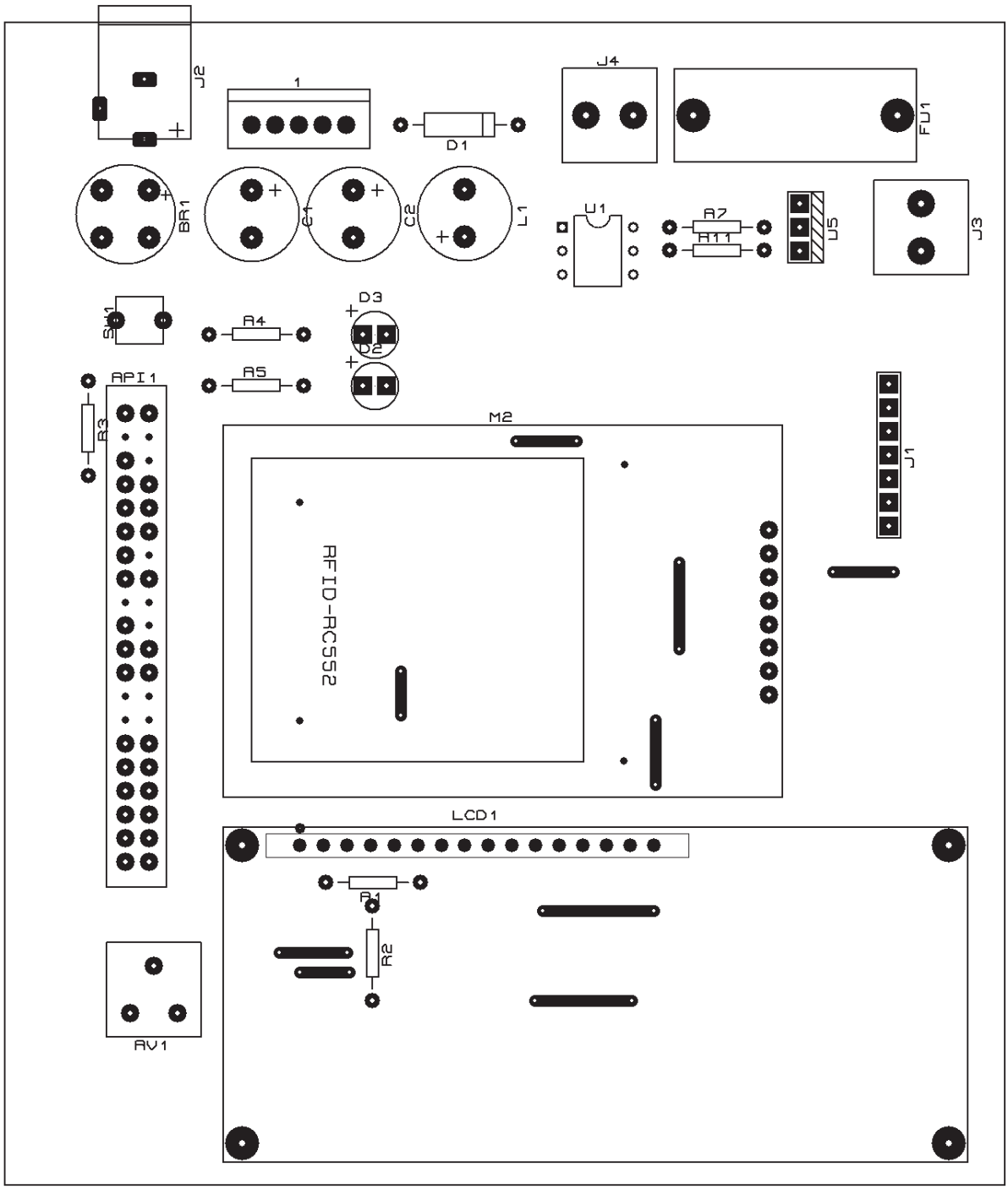
In your Python script (or in the REPL), import the GPIO module, set the board mode to that of your preference, set up the pins you want to use and turn them on:

```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)  # set board mode to Broadcom

GPIO.setup(17, GPIO.OUT)  # set up pin 17
GPIO.setup(18, GPIO.OUT)  # set up pin 18

GPIO.output(17, 1)  # turn on pin 17
GPIO.output(18, 1)  # turn on pin 18
```

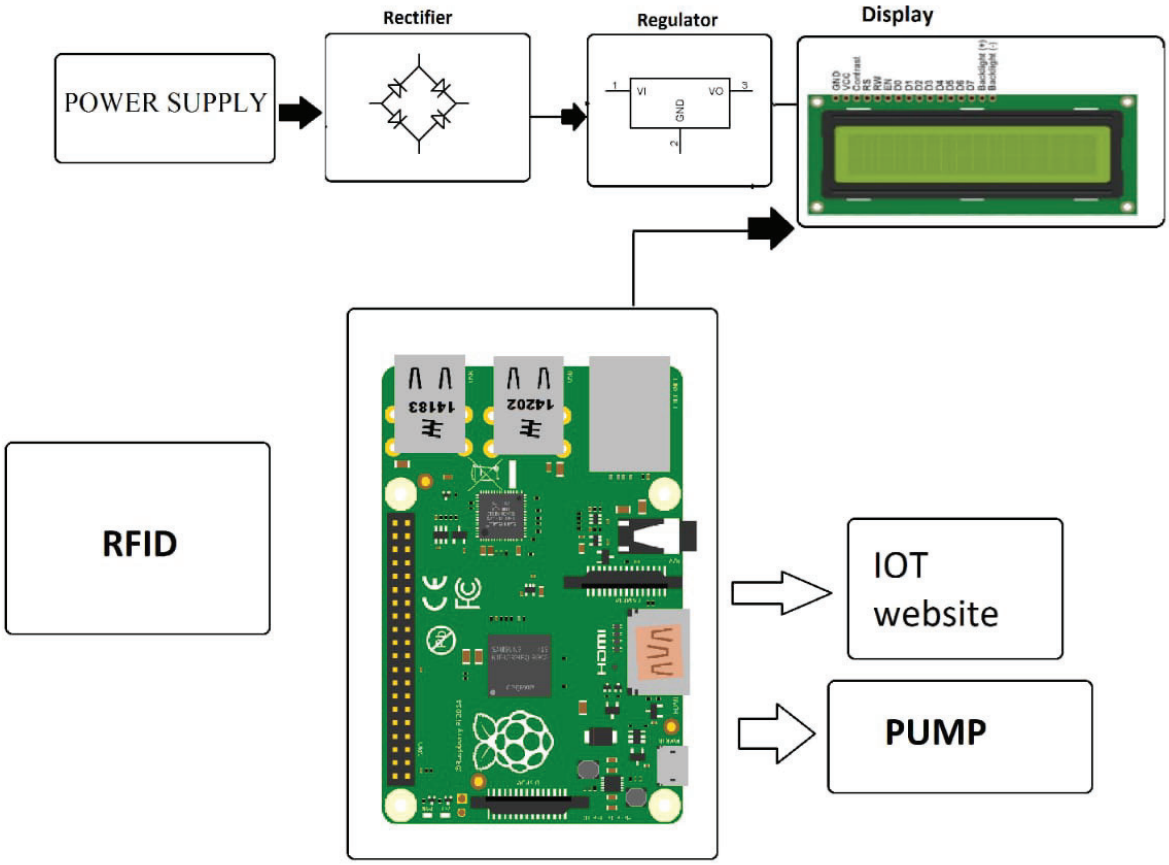
Block Diagrams



(Circuit Block Diagram)



(Extension PCB with Components Attached)



(Circuit Connection Diagram)

Chapter - 3

IMPLEMENTATION & CODING

Implementation is the stage in the project where the theoretical design is turned into the working system and is giving confidence to the new system for the users i.e. will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of method to achieve the changeover, an evaluation, of change over methods. A part from planning major task of preparing the implementation is education of users. The more complex system is implemented, the more involved will be the system analysis and design effort required just for implementation. An implementation coordinating committee based on policies of individual organization has been appointed. The implementation process begins with preparing a plan for the implementation for the system. According to this plan, the activities are to be carried out, discussions may regarding the equipment has to be acquired to implement the new system Implementation is the final and important phase. The most critical stage is in achieving a successful new system and in giving the users confidence that the new system will work and be effective. The system can be implemented only after thorough testing is done and if it found to working according to the specification. This method also offers the greatest security since the old system can take over if the errors are found or inability to handle certain types of transaction while using the new system.

The major elements of implementation plan are test plan, training plan, equipment installation plan, and a conversion plan. There are three types of implementation:

1. Implementation of a computer system to replace a manual system.
2. Implementation of a new computer system to replace an existing system.
3. Implementation of a modified application to replace an existing one, using the same computer.
4. Successful implementation may not guarantee improvement in the organization using the new system, but improper installation will prevent it.

It has been observed that even the best system cannot show good result if the analysts managing the implementation do not attend to every important detail. This is an area where the systems analysts need to work with utmost care.

Implementation Phase

1. Training personnel
2. Conversion Procedures
3. Post-implementation review

Training of Personal Involved With System

Even well designed system can succeed or fail because of the way they are operated and used. Therefore, the quality of training received by the personal involved with the system in various capacities helps or hinders and may even prevent the successful implementation of management information system.

System Operators Training

Running of the system successfully depend on the personnel working in the Computer Centre. They are Responsible for providing the necessary support. Their training must ensure that they are able to handle all possible operations, both routine and extra-ordinary in nature. If the system calls for the installation of new equipment, such as new computer system, special terminals or different data entry machines, the operators training should include such fundamentals as how to turn the equipment on and use it, how to power off and a knowledge of what constitutes normal operations. The operators should also be trained on different type of malfunctioning, how to recognize them and what steps should also be taken whenever they arise.

User Training

User may be trained on use equipment, particularly in the case where, e.g. a microcomputer is in use and individual involved is both operator and user. In such cases, user must be given training on how to operate and user. In such cases, user must be given training on how to operator the system also. Questions that may be trivial to the analyst, such as how to turn on a terminal, how to insert a diskette into a micro-computer or when it is safe to turn off equipment without danger of data loss are significant problems to new users who are not familiar.

In most of the cases user training deals with the operation of the system itself, with proper attention given to data handling techniques. It is imperative that users be properly trained in methods of entering transaction, editing data, formulating inquiries, deleting and inserting of records. No training is complete without familiarizing users with simple systems maintenance activities. Weakness in any aspect of training may lead of awkward situation that creates user frustration and error.

Conversion Methods

A conversion is the process of changing from the old system to the new one. It must be properly planned and executed. Four methods are common in use. They are Parallel Systems, Direct Conversion, Pilot System and Phase In method. Each method should be considered in the light of the opportunities that it offers and problems that it may create. In general, system conversion should be accomplished in shortest possible time. Long conversion periods create problems for all persons involved including both analysts and users.

Parallel System

The most secure method of converting from an old to new system is to run both systems in parallel. This method is safest one because it ensures that in case of any problem in using new system, the organization can still fall back to the old system without the loss of time and money.

The Disadvantages Of Parallel Systems Approach Are:

1. It doubles operating costs.
2. The new system may not get fair trial.

Direct Conversion

This method converts from the old system to new system abruptly, sometimes over a weekend or even overnight. The old system is used until a planned conversion day, when it is replaced by the new system.

Pilot System

Pilot approach is often preferred in the case of the new system which involves new techniques or some drastic changes in organization performance. In this method, a working version of the system is implemented in one part of the organization, such as a single work area or department.

Phase –In- Method

This method is used when it is not possible to install a new system throughout an organization all at once. The conversion of files, training of personnel or arrival of equipment may force the staging of the implementation over a period of time, ranging from weeks to months.

Post Implementation Review

After the system is implemented and conversion is complete, a review should be conducted to determine whether the system is meeting expectations and where improvements are needed. A post implementation review measures the systems performance against predefined requirement. It determines how well the system continues to meet the performance specifications.

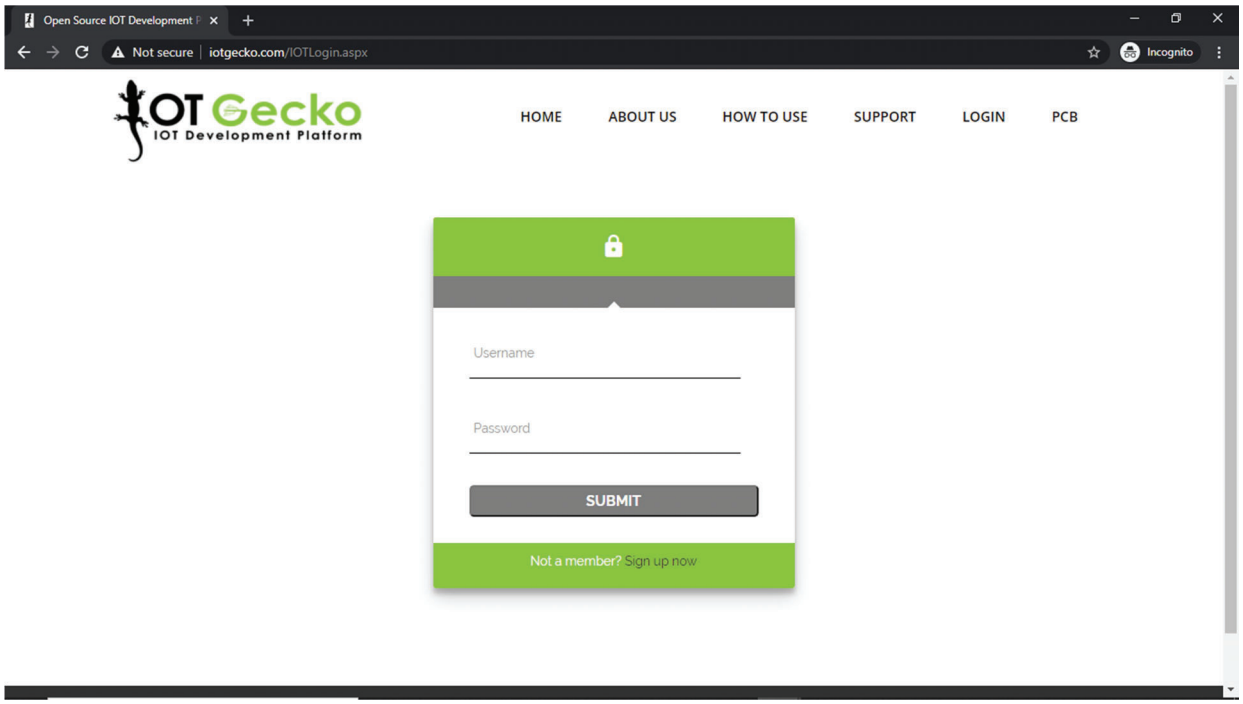
Access Procedure and Screenshots of IOT Server

Access Procedure

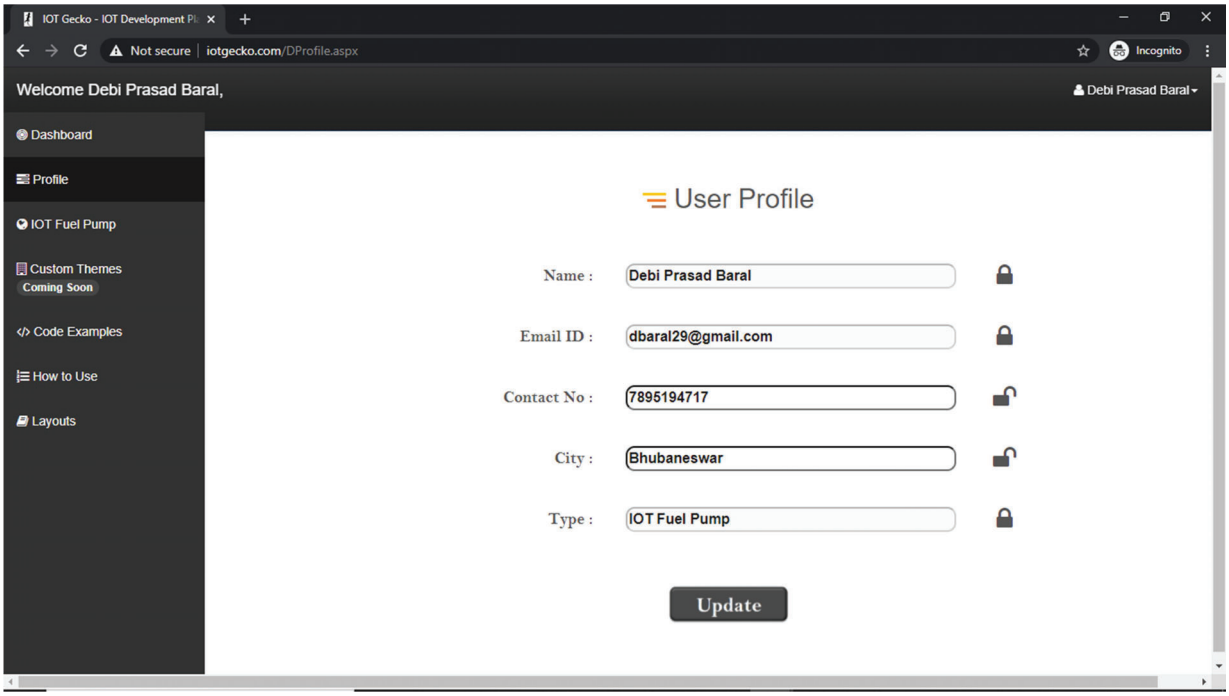
- Open the Browser and in the URL bar enter the given URL i.e. " http://iotgecko.com/IOTLogin.aspx"
- It will ask for developer Email and Password to login
- After Login it will take you to the developer profile
- In-order to access the IOT Petrol Pump we have to click on the option that says "IOT Petrol Pump"
- After that the server will ask to login to the check the details of transaction
- Here you can login as User or as Admin.
- After login server will show all the transactions done by that user if you are logged in as user, if you are logged in as Admin, it will show all the transactions done by all the users
- If the user or Admin wants to add balance in RFID card, there is a button provided in the upper right corner, click on the button and enter the amount you want to add and click on submit button
- After Successful addition of balance an alert will show up i.e. "Balance Added Successfully"
- After everything is done the user or admin can logout by clicking on the logout button

P.S - (All the Procedure Screenshots are provided in Next Few Pages)

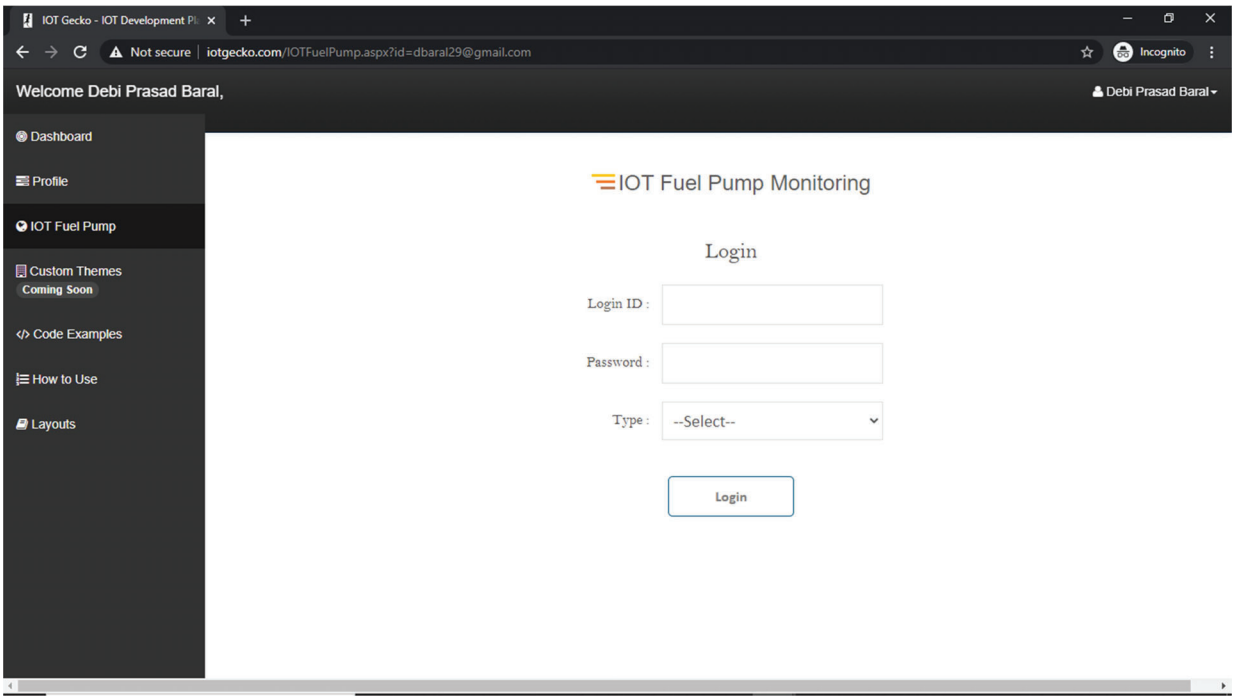
Screenshots of IOT Server



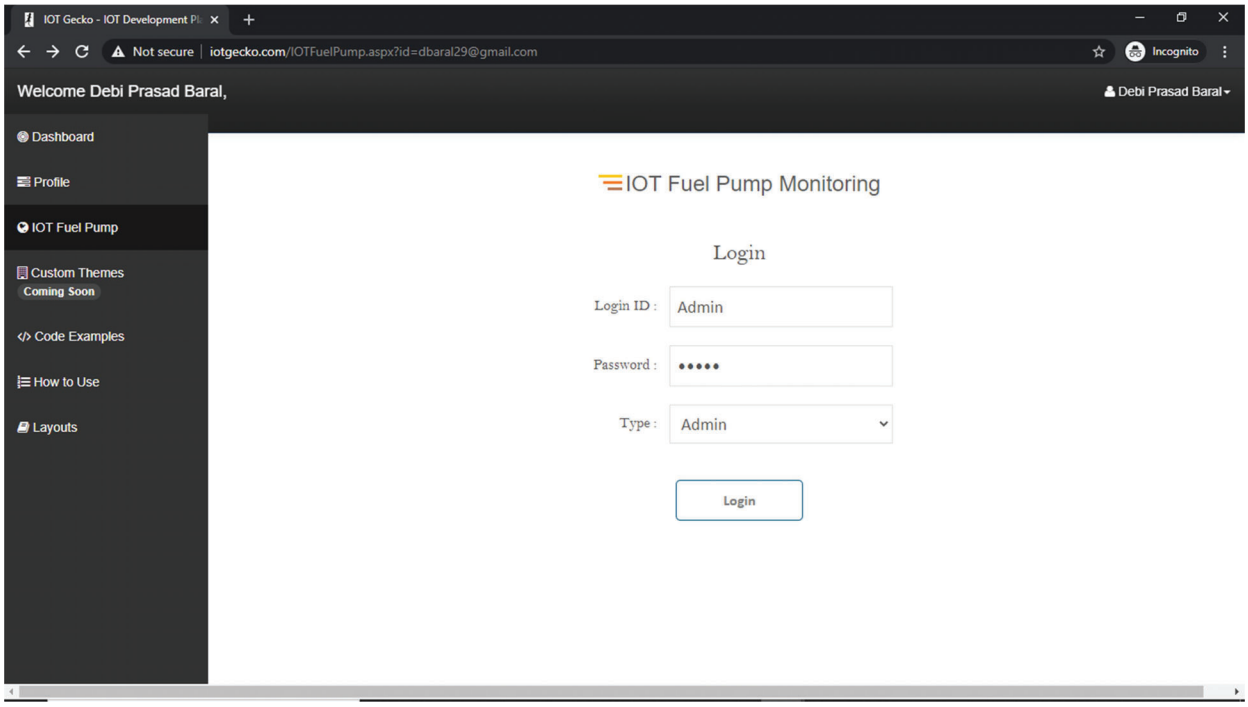
(IOT Server Login Page - http://iotgecko.com/IOTLogin.aspx)



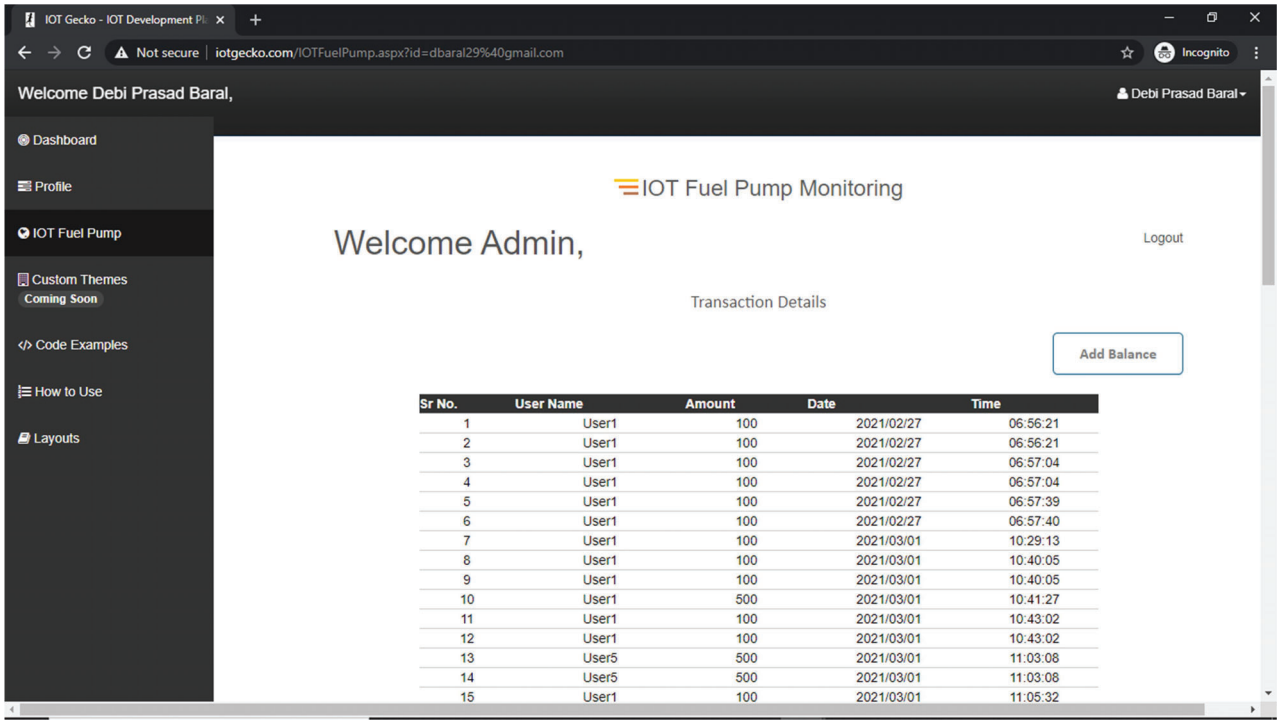
(IOT Server User Profile)



(IOT Server Admin and User Login Page)

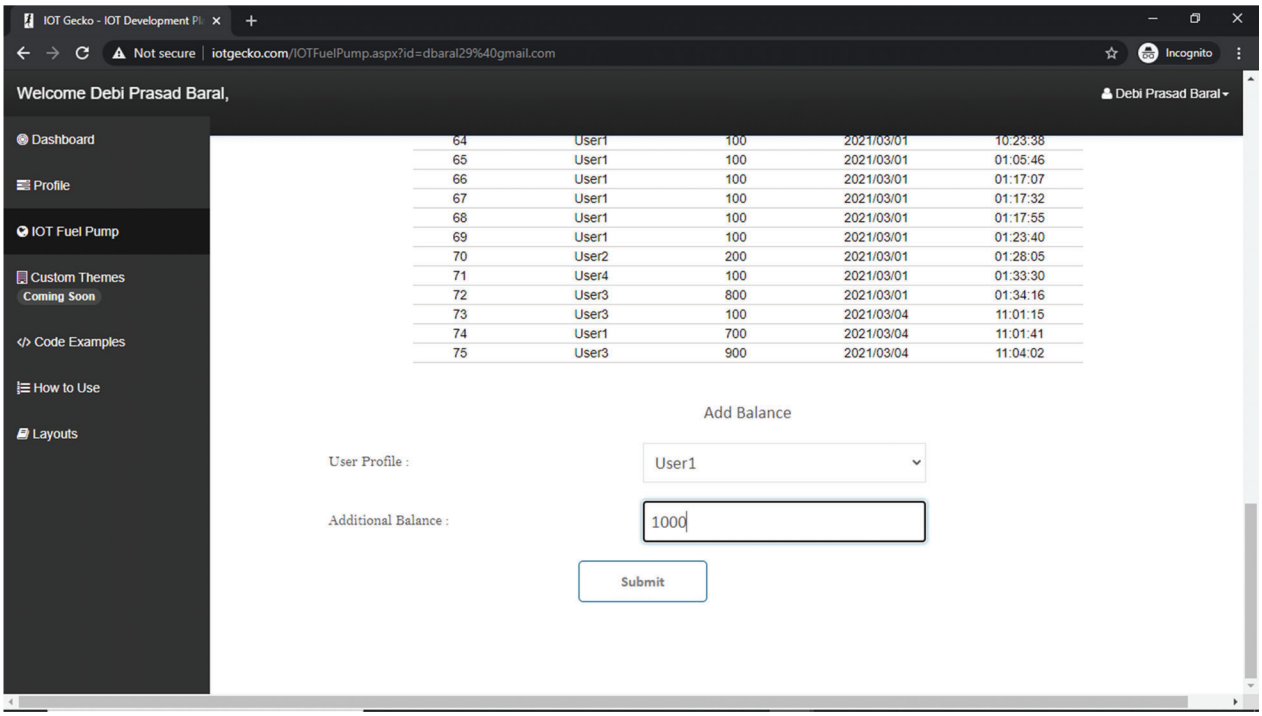


(IOT Server Admin Login)

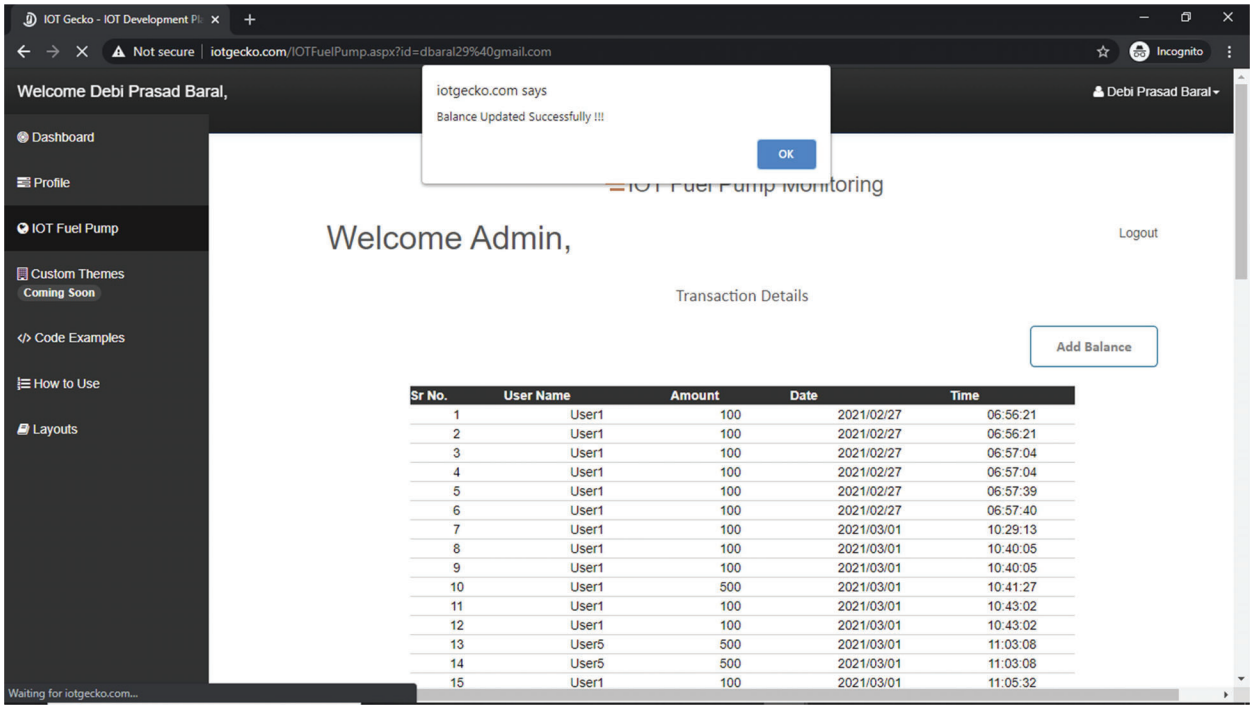


Sr No.	User Name	Amount	Date	Time
1	User1	100	2021/02/27	06:56:21
2	User1	100	2021/02/27	06:56:21
3	User1	100	2021/02/27	06:57:04
4	User1	100	2021/02/27	06:57:04
5	User1	100	2021/02/27	06:57:39
6	User1	100	2021/02/27	06:57:40
7	User1	100	2021/03/01	10:29:13
8	User1	100	2021/03/01	10:40:05
9	User1	100	2021/03/01	10:40:05
10	User1	500	2021/03/01	10:41:27
11	User1	100	2021/03/01	10:43:02
12	User1	100	2021/03/01	10:43:02
13	User5	500	2021/03/01	11:03:08
14	User5	500	2021/03/01	11:03:08
15	User1	100	2021/03/01	11:05:32

(IOT Server Admin Page)

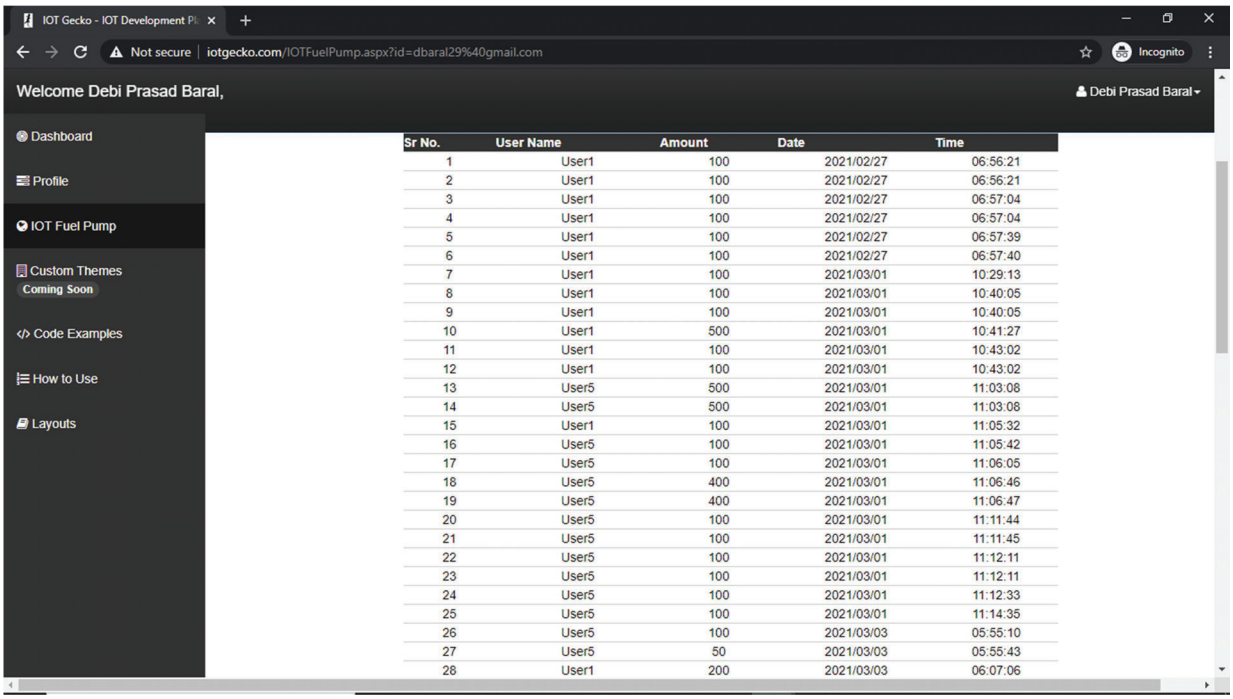


(IOT Server Admin Add Balance)



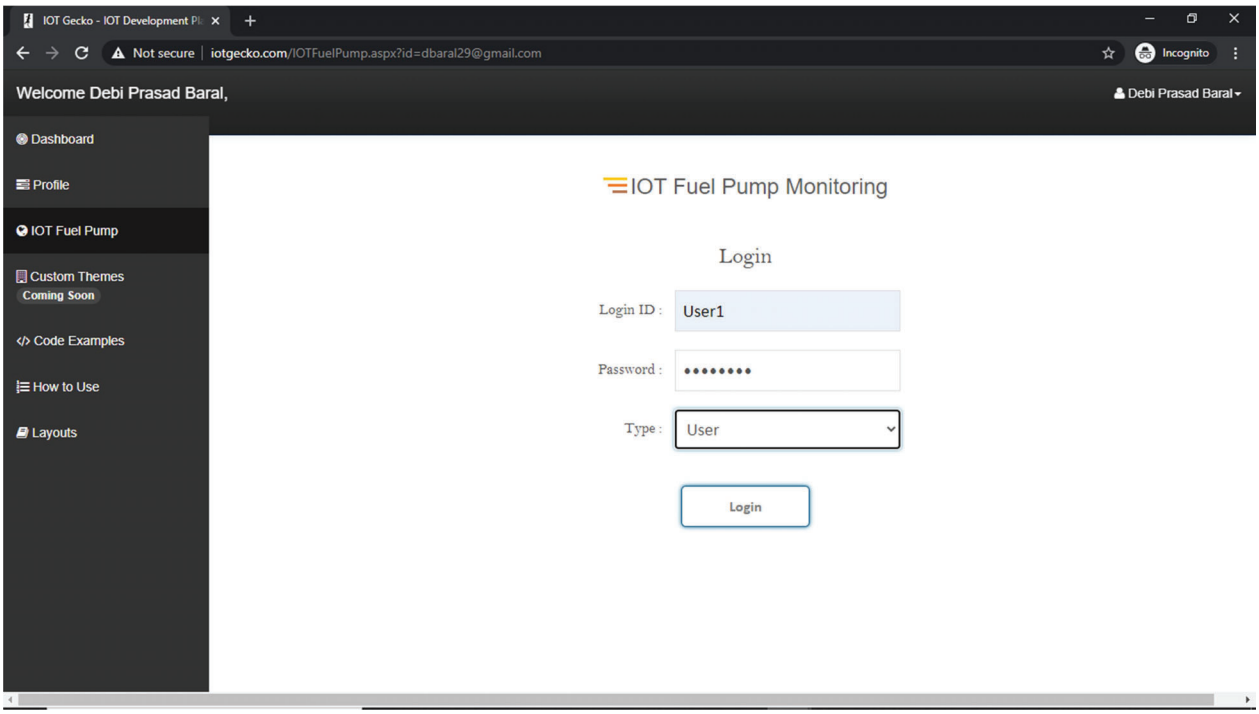
Sr No.	User Name	Amount	Date	Time
1	User1	100	2021/02/27	06:56:21
2	User1	100	2021/02/27	06:56:21
3	User1	100	2021/02/27	06:57:04
4	User1	100	2021/02/27	06:57:04
5	User1	100	2021/02/27	06:57:39
6	User1	100	2021/02/27	06:57:40
7	User1	100	2021/03/01	10:29:13
8	User1	100	2021/03/01	10:40:05
9	User1	100	2021/03/01	10:40:05
10	User1	500	2021/03/01	10:41:27
11	User1	100	2021/03/01	10:43:02
12	User1	100	2021/03/01	10:43:02
13	User5	500	2021/03/01	11:03:08
14	User5	500	2021/03/01	11:03:08
15	User1	100	2021/03/01	11:05:32

(IOT Server Admin Balance Add Successful)

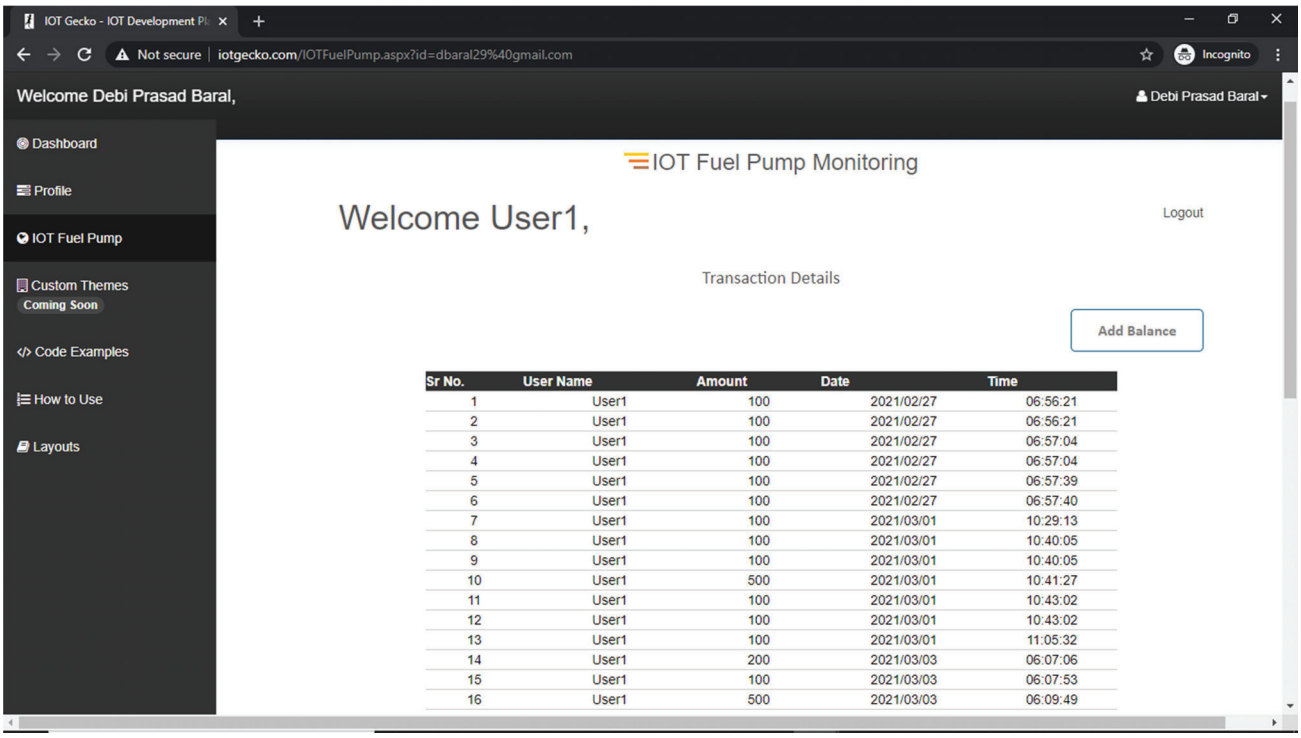


Sr No.	User Name	Amount	Date	Time
1	User1	100	2021/02/27	06:56:21
2	User1	100	2021/02/27	06:56:21
3	User1	100	2021/02/27	06:57:04
4	User1	100	2021/02/27	06:57:04
5	User1	100	2021/02/27	06:57:39
6	User1	100	2021/02/27	06:57:40
7	User1	100	2021/03/01	10:29:13
8	User1	100	2021/03/01	10:40:05
9	User1	100	2021/03/01	10:40:05
10	User1	500	2021/03/01	10:41:27
11	User1	100	2021/03/01	10:43:02
12	User1	100	2021/03/01	10:43:02
13	User5	500	2021/03/01	11:03:08
14	User5	500	2021/03/01	11:03:08
15	User1	100	2021/03/01	11:05:32
16	User5	100	2021/03/01	11:05:42
17	User5	100	2021/03/01	11:06:05
18	User5	400	2021/03/01	11:06:46
19	User5	400	2021/03/01	11:06:47
20	User5	100	2021/03/01	11:11:44
21	User5	100	2021/03/01	11:11:45
22	User5	100	2021/03/01	11:12:11
23	User5	100	2021/03/01	11:12:11
24	User5	100	2021/03/01	11:12:33
25	User5	100	2021/03/01	11:14:35
26	User5	100	2021/03/03	05:55:10
27	User5	50	2021/03/03	05:55:43
28	User1	200	2021/03/03	06:07:06

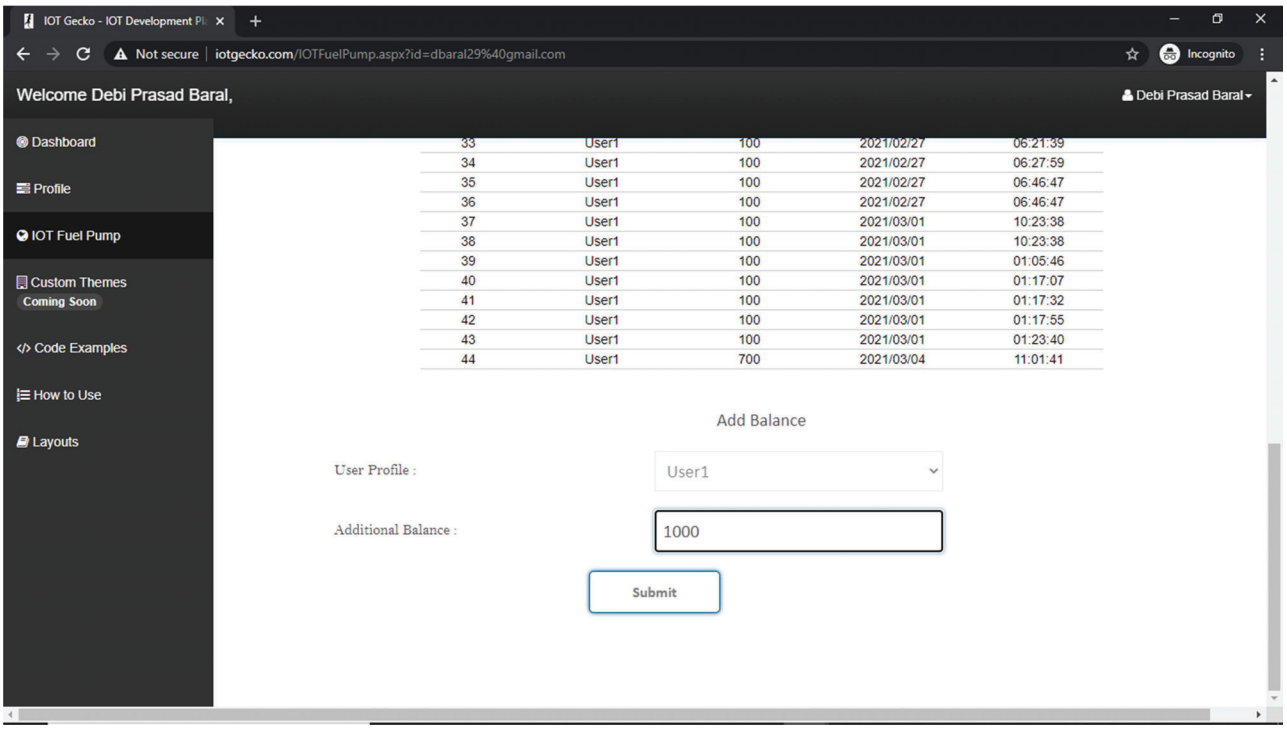
(IOT Server All User Transaction View from Admin Panel)



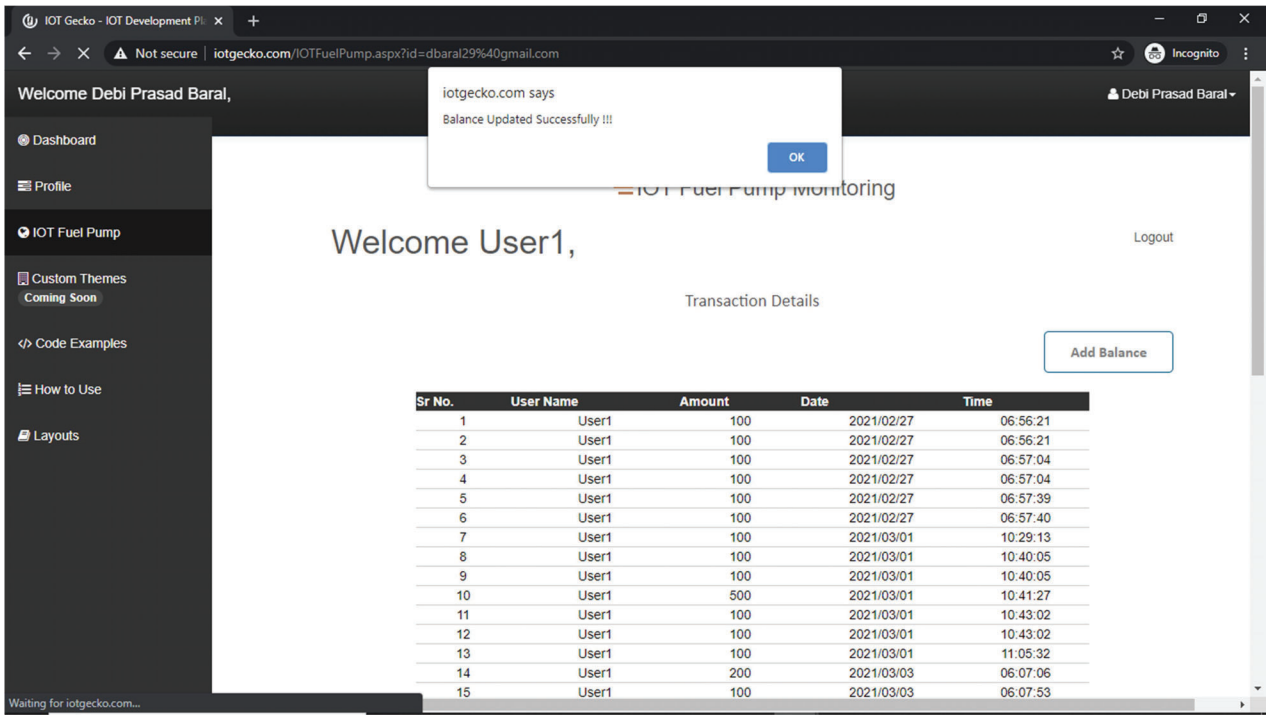
(IOT Server User Login Page)



(IOT Server User Panel View)



(IOT Server User Add Balance)



Sr No.	User Name	Amount	Date	Time
1	User1	100	2021/02/27	06:56:21
2	User1	100	2021/02/27	06:56:21
3	User1	100	2021/02/27	06:57:04
4	User1	100	2021/02/27	06:57:04
5	User1	100	2021/02/27	06:57:39
6	User1	100	2021/02/27	06:57:40
7	User1	100	2021/03/01	10:29:13
8	User1	100	2021/03/01	10:40:05
9	User1	100	2021/03/01	10:40:05
10	User1	500	2021/03/01	10:41:27
11	User1	100	2021/03/01	10:43:02
12	User1	100	2021/03/01	10:43:02
13	User1	100	2021/03/01	11:05:32
14	User1	200	2021/03/03	06:07:06
15	User1	100	2021/03/03	06:07:53

(IOT Server User Balance Add Successful)

CODING

```
#!/usr/bin/env python

import RPi.GPIO as GPIO
from mfrc522 import SimpleMFRC522
import Adafruit_CharLCD as LCD
from urllib.request import urlopen
from datetime import datetime
from smbus import SMBus
import math
import os, errno
import time
import subprocess
import requests

GPIO.setwarnings(False)
GPIO.cleanup()

site = 'http://www.iotgecko.com/IOTHit.aspx?Id=dbaral29@gmail.com&Pass=4542&Data='

# Raspberry Pi pin configuration:
lcd_rs        = 21  # Note this might need to be changed to 21 for older revision Pi's.
lcd_en        = 20
lcd_d4        = 16
lcd_d5        = 26
lcd_d6        = 19
lcd_d7        = 13

# Define LCD column and row size for 16x2 LCD.
lcd_columns = 16
lcd_rows    = 2
```

```
lcd = LCD.Adafruit_CharLCD(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7,
lcd_columns, lcd_rows)

pump = 3
button = 4
main = True

GPIO.setup(button, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(pump, GPIO.OUT)
GPIO.output(pump, True)

reader = SimpleMFRC522()

def check_connectivity():
try:
r = requests.get(site)
response = str(r.content)
print(str(response))
print(response[577:582])
return (str(response[577:582]))
except:
return False

def get_key():
kp = keypad()
digit = None
while digit == None:
digit = kp.getKey()
# Print result
#print (digit)
return digit
```

```
def get_ID():
id, text = reader.read()
print(str(id))
lcd.message(str(id))
if id == 1062576087691:
return 1
elif id == 768969218688:
return 2
elif id == 574225760880:
return 3
elif id == 32437330562:
return 4
elif id == 308071129831:
return 5

class keypad():
def __init__(self, columnCount = 3):
GPIO.setmode(GPIO.BCM)

# CONSTANTS
if columnCount is 3:
self.KEYPAD = [
[1,2,3],
[4,5,6],
[7,8,9],
["*",0,"#"]
]

self.ROW         = [14,15,18,27]
self.COLUMN      = [23,22,12]
```

```
elif columnCount is 4:
self.KEYPAD = [
[1,2,3,"A"],
[4,5,6,"B"],
[7,8,9,"C"],
["*",0,"#","D"]
]

self.ROW         = [18,23,24,25]
self.COLUMN      = [4,17,22,21]
else:
return

def getKey(self):

# Set all columns as output low
for j in range(len(self.COLUMN)):
GPIO.setup(self.COLUMN[j], GPIO.OUT)
GPIO.output(self.COLUMN[j], GPIO.LOW)

# Set all rows as input
for i in range(len(self.ROW)):
GPIO.setup(self.ROW[i], GPIO.IN, pull_up_down=GPIO.PUD_UP)


# Scan rows for pushed key/button
# A valid key press should set "rowVal"  between 0 and 3.
rowVal = -1
for i in range(len(self.ROW)):
tmpRead = GPIO.input(self.ROW[i])
if tmpRead == 0:
rowVal = i
```

```
# if rowVal is not 0 thru 3 then no button was pressed and we can exit
if rowVal <0 or rowVal >3:
self.exit()
return

# Convert columns to input
for j in range(len(self.COLUMN)):
GPIO.setup(self.COLUMN[j], GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

# Switch the i-th row found from scan to output
GPIO.setup(self.ROW[rowVal], GPIO.OUT)
GPIO.output(self.ROW[rowVal], GPIO.HIGH)

# Scan columns for still-pushed key/button
# A valid key press should set "colVal"  between 0 and 2.
colVal = -1
for j in range(len(self.COLUMN)):
tmpRead = GPIO.input(self.COLUMN[j])
if tmpRead == 1:
colVal=j

# if colVal is not 0 thru 2 then no button was pressed and we can exit
if colVal <0 or colVal >2:
self.exit()
return

# Return the value of the key pressed
self.exit()
return self.KEYPAD[rowVal][colVal]
```

```
def exit(self):
# Reinitialize all rows and columns as input at exit
for i in range(len(self.ROW)):
GPIO.setup(self.ROW[i], GPIO.IN, pull_up_down=GPIO.PUD_UP)
for j in range(len(self.COLUMN)):
GPIO.setup(self.COLUMN[j], GPIO.IN, pull_up_down=GPIO.PUD_UP)


amount = []

while GPIO.input(button) == True:
lcd.clear()
lcd.message(" IoT based RFID \n   Petrol Pump  ")
time.sleep(2)
lcd.clear()
lcd.message("     using      \n  Raspberry Pi  ")
time.sleep(2)
lcd.clear()
lcd.message('Connecting to \nInternet...')
time.sleep(2)
t1 = datetime.now()

while not check_connectivity():
t2 = datetime.now()
delta = t2 - t1
time_elapse = delta.total_seconds()
if time_elapse > 10:
lcd.clear()
lcd.message('Error: Check\nyour Internet!')
main = False
lcd.clear()
lcd.message("Press 'RESET' to\nRestart")
```

```
while GPIO.input(button) == True:
None
break
else:
main = True




lcd.clear()
lcd.message('Connected!')
time.sleep(2)
lcd.clear()
lcd.message("Scan your Card:\n")

t1 = datetime.now()
while main == True:
print('ok')
user = get_ID()
print(user)
time.sleep(2)
lcd.clear()
lcd.message("User " + str(user) + " detected!")
time.sleep(2)


lcd.clear()
lcd.message("Enter amount: \n")
for i in range(3):
amount.append(str(get_key()))
lcd.set_cursor(i, 1)
lcd.message(str(get_key()))
time.sleep(0.5)
```

```
lcd.clear()
lcd.message("Please wait...")
site = 'http://www.iotgecko.com/IOTHit.aspx?Id=dbaral29@gmail.com&Pass=4542&Data=' + str(user) + '*' +
str(amount[0]) + str(amount[1]) + str(amount[2])
getResponse = check_connectivity()
if(getResponse.startswith('0') == True):
lcd.clear()
lcd.message("Sorry!! \nLow Balance")
time.sleep(2)
lcd.clear()
lcd.message("Scan your Card:\n")
else:
samount = str(amount[0]) + str(amount[1]) + str(amount[2])
iamount = int(samount)
lcd.clear()
lcd.message("Pump ON!\nAmount: " + str(amount[0]) + str(amount[1]) + str(amount[2]))
if(iamount >= 0 and iamount < 100):
GPIO.output(pump, False)
time.sleep(3)
GPIO.output(pump, True)
elif(iamount >= 100 and iamount < 200):
GPIO.output(pump, False)
time.sleep(6)
GPIO.output(pump, True)
elif(iamount >= 200 and iamount < 300):
GPIO.output(pump, False)
time.sleep(9)
GPIO.output(pump, True)
elif(iamount >= 300 and iamount < 400):
GPIO.output(pump, False)
time.sleep(12)
GPIO.output(pump, True)
```

```
 elif(iamount >= 400 and iamount < 500):
GPIO.output(pump, False)
time.sleep(15)
GPIO.output(pump, True)
elif(iamount >= 500 and iamount < 600):
GPIO.output(pump, False)
time.sleep(18)
GPIO.output(pump, True)
elif(iamount >= 700 and iamount < 800):
GPIO.output(pump, False)
time.sleep(21)
GPIO.output(pump, True)
elif(iamount >= 900 and iamount < 1000):
GPIO.output(pump, False)
time.sleep(23)
GPIO.output(pump, True)
if(getResponse == "/span"):
lcd.clear()
lcd.message("Sorry!! \nLow Balance")
time.sleep(2)
lcd.clear()
lcd.message("Scan your Card:\n")
else:
text_word = getResponse.split('*')
print(text_word)
lcd.clear()
lcd.message("Balance Amount:\n" + str(text_word[1]))
time.sleep(3)

amount = []

lcd.clear()
lcd.message("Scan your Card:\n")
```

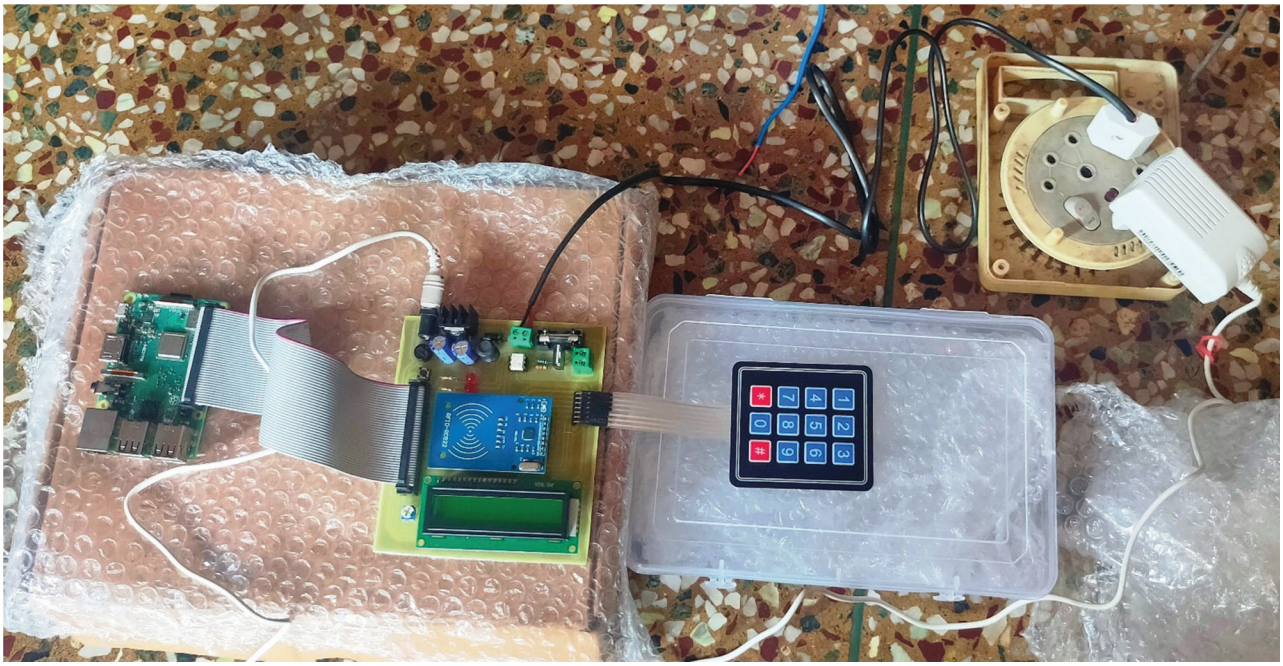
Access Procedure and Screenshots of Module

Access Procedure

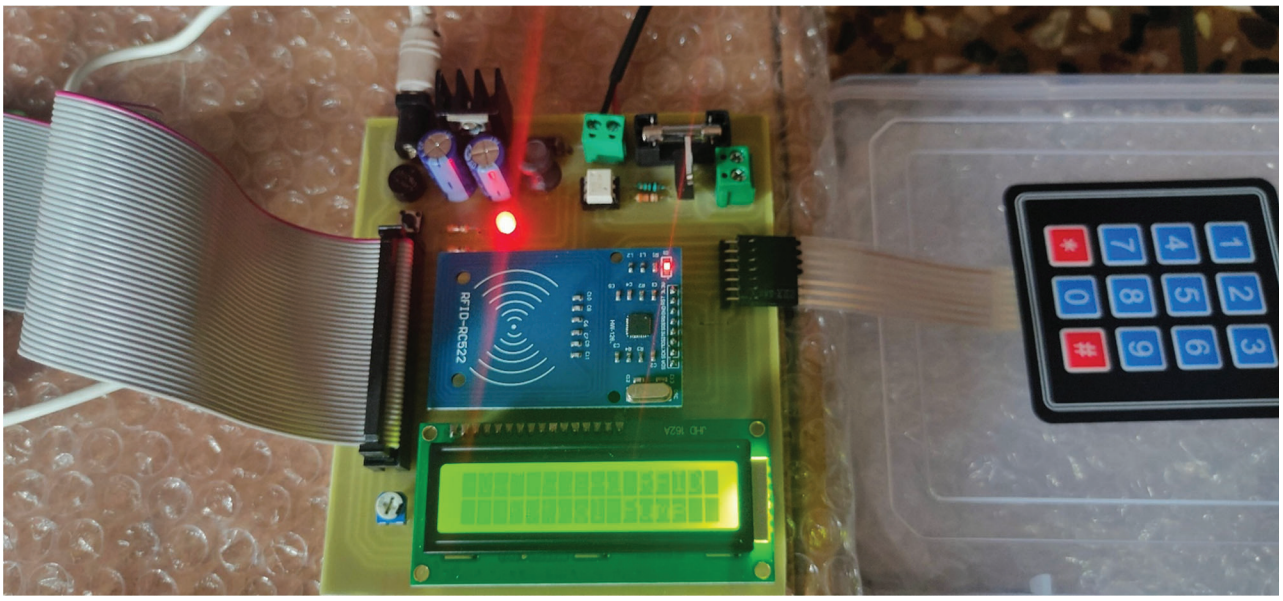
- Plug in the Power cables and turn on the switch to supply power to the module
- It will show up alert texts on the display i.e. ” IOT Petrol Pump ”
- Then it will ask to scan the RFID card.
- When user will scan the card, it will show another alert text i.e. “User Detected”
- In the next step it will ask to enter the amount of fuel you want to buy
- When the user will enter the amount the microcontroller will the take the amount as input and it will automatically start the pump and a text will also show up i.e. “Pump On”
- After the filling of fuel it will display the remaining balance in the card
- At the end the system will return to the initial stage and will ask to scan the RFID card

P.S - (All the Procedure Screenshots are provided in Next Few Pages)

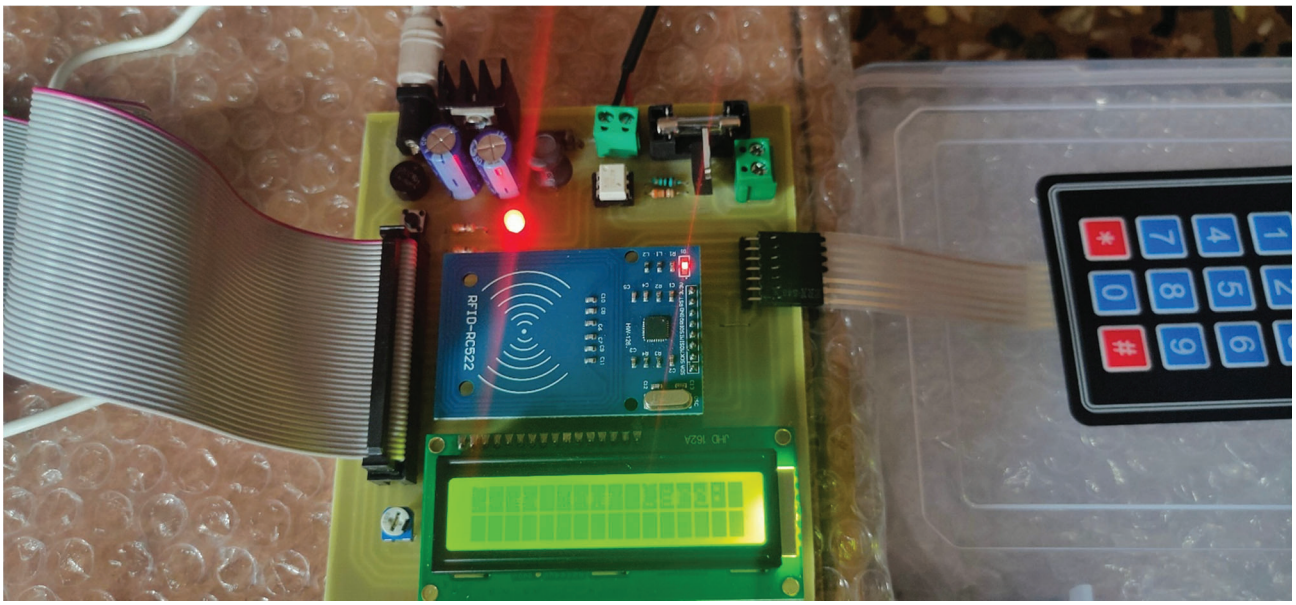
Screenshots of Module and Access Procedure



(Plug in the Power cables and turn on the switch)

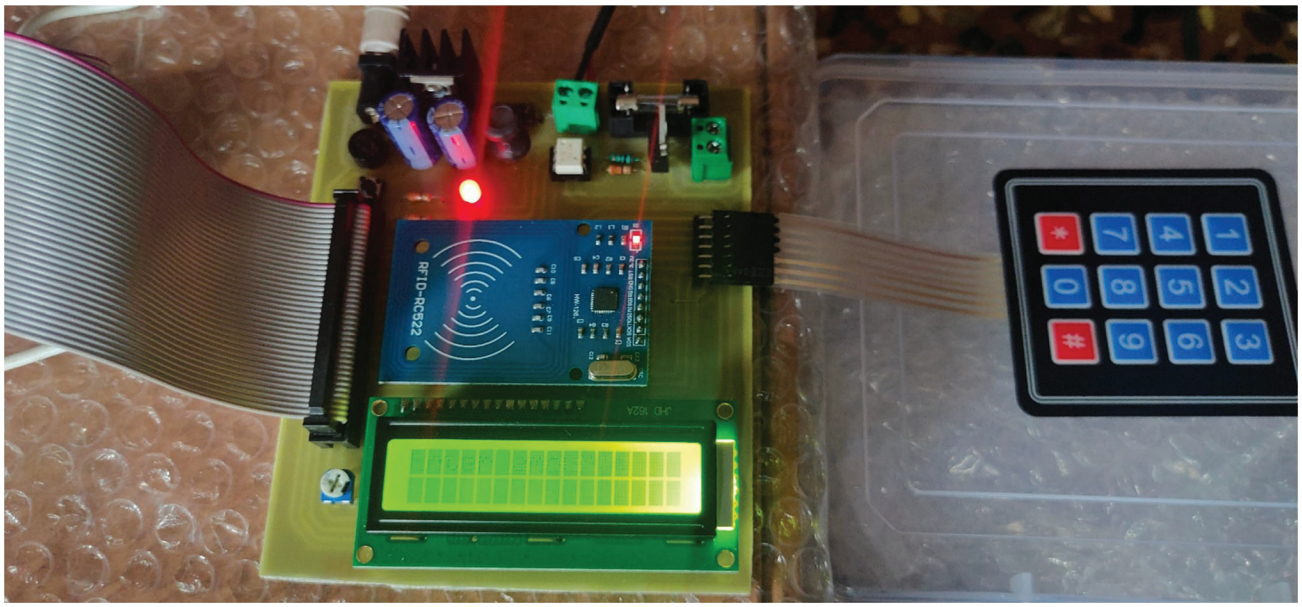


(Power on)



(Scan Your Card)

(User Detected)



(Enter Amount)

(Pump is on after entering the amount)

(Pump Filling Fuel)

(Showing the Remaining Balance)

CHAPTER 4

TESTING & TESTING RESULTS

SYSTEM TESTING

System testing is a critical element of quality assurance and represents the ultimate review of analysis, design and coding. Test case design focuses on a set of techniques for the creation of test because that meet overall testing objective. When a system is developed it is hoped that it performs properly. The main purpose of testing an information system is to find the errors and correct them. The scope of system testing should include both manual and computerized operations. System testing is comprehensive evaluation of the programs, manual procedures, computer operations and controls. System testing is the process of checking whether the developed system is working according to the objective and requirement. All testing is to be conducted in accordance to the test conditions specified earlier. This will ensure that the test coverage meets the requirements and that testing is done in a systematic manner.

TEST CHARACTERS

1. A good test is not redundant.
2. A good a good test has a high probability of finding an error.
3. Test should be ―best of breed‖.

Levels of Testing

Requirements -System Testing

Design- User Need -Acceptance Testing

Integration Testing

Code- Unit Testing

BLACK BOX TESTING:

The method of Black Box Testing is used by the software engineer to derive the required results of the test cases:

1. Lack Box Testing alludes to test that are conducted at the software interface.
2. Black Box Test examines some fundamental aspect of a system with little regard for the internal logic structure of the software.
3. A limited number of important logical paths can be selected and exercised. Black box testing was performed to find errors in the following categories: - Incorrect or missing functions

1. Graphics error.
2. Errors in data in binary format.
3. Error in data in integer format.
4. File error.
5. Pointer error
6. Variable error

WHITE BOX TESTING:

White Box Testing is sometimes called Glass Box Testing. Using White Box Testing methods the software engineer can derive the following test cases:

1. Guarantee that all independent paths within a module have been exercised at least once.
2. Exercise all logical decisions on their true and false sides.
3. Execute all loops at their boundaries and within their operational bounds.

4. In White Box Testing efforts were made to handle the following:-

- Number of input parameters equal to number of arguments.
- Parameters and arguments attributes match.
- Number of arguments transmitted is called modules equal to attributes of parameters.
- Unit system of argument transmitted is called modules equal unit system of parameter.
- Number of attributes and order of arguments to build in functions correct.
- Any references to parameters not associated to build in functions correct.
- Input only arguments altered.
- Global variable definition consistent across module.
- Files attributes correct.

Unit Testing

Code testing was carried out to see the correctness of the logic involved and the correctness of the modules. Tests were conducted based upon sample and live data as well. All the modules are checked separately for assuming correctness and accuracy in all the calculations.

Specification Testing

It examines the specification stating about what program should do and how it performs under various conditions. This testing strategy is better strategy since it focuses on the way the software is expected to work.

Integration Testing

The next level testing that was performed is often referred to as integration testing. During this phase many unit tested modules were combined into subsystems, which were then tested. The goal here was to see if modules can be integrated properly. Here the emphasis was on testing interfaces between different constituent modules of system.

Functionality Testing

Here the entire software system was tested. The reference document for this process is the requirements document, and the goal was to see if software solution meets its requirements. This level of testing is essentially a validation exercise, and in many situation it is the only validation activity.

Stress Testing

Proxy server developed for the specified purpose was testing under heavy load, i.e. a large no. of clients were made to sit in lab and were asked to send requests for logging in and then were asked to request for text on internet. System responded to request as desired.

Acceptance Testing

Acceptance was performed in the real environment with realistic data of the client to demonstrate if the software developed is working satisfactorily. Here the main focus was on the external behavior of the system; the internal logic of the program was not emphasized.

Test Data and Test Cases

The primary objective of test case design is to derive a set of tests that have the highest likelihood of uncovering errors in software. The test case specification is the major activity in the testing process. Careful selection of test cases that satisfy the criterion on approach specified is essential for proper testing. Various characteristics of test cases that are required for portal are:

1. A good test has a high probability of finding an error.
2. A good test is not redundant.
3. A good test should be ―Best of Breed.
4. A good test should be neither too simple not too complex.

Overview of Testing

Testing: Testing involves executing the program (or part of it) using sample data and inferring from the output whether the software performs correctly or not. This can be done either during module development (unit testing) or when several modules are combined (system testing).

2. **Direct Testing**: Defect testing is testing for situation where the program does not meet its functional specification. Performance testing tests a system's performance or reliability under realistic loads. This may go some way to ensuring that the program meets its nonfunctional requirements.

Debugging

Debugging is a cycle of detection, location, repair and test. Debugging is a hypothesis testing process. When a bug is detected, the tester must form a hypothesis about the cause and location of the bug. Further examination of the execution of the program (possible including many returns of it) will usually take place to confirm the hypothesis. If the hypothesis is demonstrated to be incorrect, a new hypothesis must be formed. Debugging tools that show the state of the program are useful for this, but inserting print statements is often the only approach. Experienced debuggers use their knowledge of common and/or obscure bugs to facilitate the hypothesis testing process. After fixing a bug, the system must be reset to ensure that the fix has worked and that no other bugs have been introduced. This is called regression testing. In principle, all tests should be performed again but this is often too expensive to do.

Test Planning

Testing needs to be planned, to be cost and time effective. Planning is setting out standards for tests. Test plans set out the context in which individual engineers can place their own work. Typical test plan contains:

1. **Parameter**: data (or occasionally function references) passed from one unit to another.
2. **Shared memory**: block of memory shared between units (e.g. global variable) .One places data there and the other retrieves it.
3. **Procedural**: Object-Oriented or abstract data type form of interface, encapsulating several procedures.
4. **Message passing**: one sub-system requests a service by passing a message. Client-server interface also used by some OO architectures.

- **Interface Testing**: Usually done at integration stage when modules or sub-systems are combined. Objective is to detect errors or invalid assumptions about interfaces between modules. Reason these are not shown in unit testing is that test case may perpetuate same incorrect assumptions made by module designer. Particularly important when OO development has been used.

Four Types of Interface

Typical levels of testing:

Acceptance testing - whole system with real data (involve customer, user, etc) .Alpha testing is acceptance testing with a single client (common for bespoke systems). Beta testing involves distributing system to potential customers to use and provide feedback. In, this project, Beta testing has been followed. This exposes system to situations and errors that might not be anticipated by us.

Maintenance

Once the website is launched, it enters the maintenance phase. All systems need maintenance. Maintenance is required because there are often some residual errors remaining in the system that must be removed as they are discovered. Maintenance involves understanding the effects of the change, making the changes to both the code and the documents, testing the new parts and retesting the old parts that were not changed.

Maintenance is mainly of two types:

1. corrective Maintenance
2. Adaptive Maintenance

Corrective Maintenance

Almost all software that is developed has residual errors or bugs in them. Many of these surfaces only after the system have been in operation, sometimes for a long time. These errors once discovered need to be removed, leading to the software to be changed. This is called Corrective Maintenance.

Adaptive Maintenance

Even without bugs, software frequently undergoes change. The software often must be upgraded and enhanced to include more features and provide more services.

Perfective Maintenance

This includes modifications and updates done in order to keep the software usable over long period of time. It includes new features, new user requirements for refining the software and improve its reliability and performance.

Preventive Maintenance

This includes modifications and updating to prevent future problems of the software. It aims to attend problems, which are not significant at this moment but may cause serious issues in future.

Cost of Maintenance

Reports suggest that the cost of maintenance is high. A study on estimating software maintenance found that the cost of maintenance is as high as 67% of the cost of entire software process cycle. On an average, the cost of software maintenance is more than 50% of all SDLC phases.
There are various factors, which trigger maintenance cost go high, such as:

- Real-world factors affecting Maintenance Cost

1. The standard age of any software is considered up to 10 to 15 years.
2. Older software‘s, which were meant to work on slow machines with less memory and storage capacity cannot keep themselves challenging against newly coming enhanced software‘s on modern hardware.
3. As technology advances, it becomes costly to maintain old software.
4. Most maintenance engineers are newbie and use trial and error method to rectify problem.
5. Often, changes made can easily hurt the original structure of the software, making it hard for any subsequent changes.
6. Changes are often left undocumented which may cause more conflicts in future.

- Software-end factors affecting Maintenance Cost
    1. Structure of Software Program
    2. Programming Language
    3. Dependence on external environment
    4. Staff reliability and availability

MAINTENANCE ACTIVITIES

IEEE provides a framework for sequential maintenance process activities. It can be used in iterative manner and can be extended so that customized items and processes can be included. These activities go hand-in-hand with each of the following phase:

1. **Identification & Tracing** - It involves activities pertaining to identification of requirement of modification or maintenance. It is generated by user or system may itself report via logs or error messages. Here, the maintenance type is classified also.

2. **Analysis** - The modification is analyzed for its impact on the system including safety and security implications. If probable impact is severe, alternative solution is looked for. A set of required modifications is then materialized into requirement specifications. The cost of modification/maintenance is analyzed and estimation is concluded.

3. **Design** - New modules, which need to be replaced or modified, are designed against requirement specifications set in the previous stage. Test cases are created for validation and verification.

4. **Implementation** - The new modules are coded with the help of structured design created in the design step. Every programmer is expected to do unit testing in parallel.

1. **System Testing** - Integration testing is done among newly created modules. Integration testing is also carried out between new modules and the system. Finally the system is tested as a whole, following regressive testing procedures.

2. **Acceptance Testing** - After testing the system internally, it is tested for acceptance with the help of users. If at this state, user complaints some issues they are addressed or noted to address in next iteration.

3. **Delivery** - After acceptance test, the system is deployed all over the organization either by small update package or fresh installation of the system. The final testing takes place at client end after the software is delivered. Training facility is provided if required, in addition to the hard copy of user manual.

4. **Maintenance management** - Configuration management is an essential part of system maintenance. It is aided with version control tools to control versions, semi version or patch management.

Software Re-engineering

When we need to update the software to keep it to the current market, without impacting its functionality, it is called software re-engineering. It is a thorough process where the design of software is changed and programs are re-written. Legacy software cannot keep tuning with the latest technology available in the market. As the hardware become obsolete, updating of software becomes a headache. Even if software grows old with time, its functionality does not. For example, initially Unix was developed in assembly language. When language C came into existence, Unix was re-engineered in C, because working in assembly language was difficult. Other than this, sometimes programmers notice that few parts of software need more maintenance than others and they also need re-engineering.

Re-Engineering Process

a. Decide what to re-engineer. Is it whole software or a part of it?
b. Perform Reverse Engineering, in order to obtain specifications of existing software.
c. Restructure Program if required. For example, changing function-oriented programs into object oriented programs.
d. Re-structure data as required.
e. Apply Forward engineering concepts in order to get re-engineered software.

There are few important terms used in Software re-engineering

Reverse Engineering

It is a process to achieve system specification by thoroughly analyzing, understanding the existing system. This process can be seen as reverse SDLC model, i.e. we try to get higher abstraction level by analyzing lower abstraction levels. An existing system is previously implemented design, about which we know nothing. Designers then do reverse engineering by looking at the code and try to get the design. With design in hand, they try to conclude the specifications. Thus, going in reverse from code to system specification.

Program Restructuring

It is a process to re-structure and re-construct the existing software. It is all about rearranging the source code, either in same programming language or from one programming language to a different one. Restructuring can have either source code restructuring and data-restructuring or both. Re-structuring does not impact the functionality of the software but enhance reliability and maintainability. Program components, which cause errors very frequently can be changed, or updated with re-structuring. The dependability of software on obsolete hardware platform can be removed via restructuring.

Forward Engineering

Forward engineering is a process of obtaining desired software from the specifications in hand which were brought down by means of reverse engineering. It assumes that there was some software engineering already done in the past. Forward engineering is same as software engineering process with only one difference – it is carried out always after reverse engineering

Component reusability

A component is a part of software program code, which executes an independent task in the system. It can be a small module or sub-system itself.

TESTING AND ITS OBJECTIVES

Testing is a process of executing a program with intend of finding an error. Testing is a crucial element of software quality assurance and presents ultimate review of specification, design and coding.

System Testing is an important phase. Testing represents an interesting anomaly for the software. Thus a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

A good test case is one that has a high probability of finding an as undiscovered error. A successful test is one that uncovers an as undiscovered error.

Testing Objectives:

1. Testing is a process of executing a program with the intent of finding an error

2. A good test case is one that has a probability of finding an as yet undiscovered error

3. A successful test is one that uncovers an undiscovered error.

FACING PROBLEM DURING DEVELOPMENT

We have faced some problems during development —"IOT Petrol Pump" ‖ project.
Some problems are given below briefly.

1. **Requirement Gathering Phase:** It is very important step. If the requirements are not good, the project will fail. At that time, we became disappointed when we were collecting information and data then what information and data will be helpful or appropriate for this project.
2. **During Design Phase:** In this time, we were confused for drawing diagram.
3. **Development Phase:** It is a critical part of the project. At this time, we have mistaken with some parts of coding which we debugged and corrected at last.
4. **Testing Phase:** It is necessary part of the project. This part will help to test the whole project. In this time, we face some bugs of the project.

FUTURE DEVELOPMENT

Future Development is very important for each project because it includes latest features in the system. It reduces software bugs and problems. It creates strong relationship with customer according to their feedback or choices. We will integrate some dynamic feature in my IOT Petrol Pump Project which features may integrate…

- Reporting module with real time mechanism.
- Latest design structure with seamless flow.
- E-mail & Mobile confirmation system.
- Password Protection of RFID Cards.

CHAPTER 5

CONCLUSION

After completing our project on the topic “IOT Petrol Pump” we can conclude that we have found an easier and more convenient way to help the users in meeting up their requirements.

All good things must come to an end, so does our IOT Petrol Pump. We are done with the basic transaction processes for our project. The advance version of this project can be developed which we will upgrade with our future updates.

The whole credit goes to our teachers and internet for refining our ideas and making our project successful.

THANK YOU…

BIBLIOGRAPHY

- Google for Problem Solving
- https://www.raspberrypi.org/software/
- https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
- https://docs.python.org/3/
- https://pypi.org/project/RPi.GPIO/
- https://pypi.org/project/mfrc522/
- http://iotgecko.com/IOTLogin.aspx