

**INDUSTRIAL TRAINING REPORT**  
**B.E (ECE)-8<sup>TH</sup> SEM**  
**IN**  
**INTERNET OF THINGS**  
**For project on IOT BASED HOME AUTOMATION USING**  
**RASPBERRY PI**  
**Undertaken at**  
**National Institute of Electronics & Information Technology**  
**(NIELIT)**

Submitted By:  
**RAJNEET KAUR**  
**ROLL No.-UE145071**  
**B.E ECE (SEM 8)**

Under the Guidance of  
**Dr.SARWAN SINGH**  
(Deputy Director, NIELIT)



**Department of Electronics and Communication Engineering**  
**UNIVERSITY INSTITUTE OF ENGINEERING & TECHNOLOGY**  
**PANJAB UNIVERSITY, CHANDIGARH**

## ACKNOWLEDGEMENT

*The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all through the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them. I am thankful and fortunate enough to get a chance for undergoing a training at National Institute of Electronics & Information Technology.*

*I respect and thank Dr.Sarwan Singh for providing me an opportunity to do the project work in National Institute of Electronics & Information Technology and giving me all support and guidance which made me complete the project duly. I am extremely thankful to him for providing such consistent support and guidance. I owe our deep gratitude to Mr. Manik Kalsi who also guided us all by providing all the necessary information required for the completion of this project. Also, I would like to extend my sincere esteems to Mr. Jaspreet Singh who took keen interest in my project work and for his timely support.*

I extend a heartfelt thanks to my family and friends for being a constant support throughout. Last but not the least I am thankful to those sources, persons who helped me directly or indirectly in achieving this stage of our project.

RAJNEET KAUR

## **ABSTRACT**

With every breath technological entrepreneurs take, a new idea is pitched. Keeping up with these everyday breakthroughs can be tough. In less than 10 years we have witnessed the launch of yet a new wave, Internet of Things. We see IoT as billions of smart, connected things that will encompass every aspect of our lives, and its foundation is the intelligence that embedded processing provides. The IoT is comprised of smart machines interacting and communicating with other machines, objects, environments and infrastructures. As a result, huge volumes of data are being generated, and that data is being processed into useful actions that can command and control things to make our lives much easier and safer and to reduce our impact on the environment. The creativity of this new era is boundless, with amazing potential to improve our lives.

Internet of Things is a vision that was introduced in 2009. This vision encompasses the idea of connecting all devices and gadgets to the internet. The Internet of Things is truly changing our world. It is enhancing our lives, businesses, health and society as a whole by developing products which would ease our life. Components of this breakthrough are still new and not many people are familiar with these concepts. In order to seize the opportunity, we have done some work that is focused in this area.

# TABLE OF CONTENTS

<b>ABOUT NIELIT .....</b>	<b>8</b>
<b>1. INTRODUCTION.....</b>	<b>9-13</b>
1.1 Internet Of Things	
1.1.1 IoT Definition	
1.1.2 Enabling Technologies	
1.1.3 Connecting Models	
1.1.4 Transformation Potential	
1.1.5 Security	
1.1.6 Privacy	
1.2 Components Of IoT	
1.2.1 Sensors and actuators	
1.2.2 Connectivity	
1.2.3 People and Processes	
<b>2. CHAPTER 2( Work Done)(Jan-march).....</b>	<b>14-20</b>
2.1 Familiarization with C language and Data Structures	
2.2 Proteus	
2.3 Keil	
2.4 8051 microprocessor	
2.5 ARM 7	
<b>3. CHAPTER 3(Work Done)(April-June).....</b>	<b>21-29</b>
3.1 Arduino Uno	
3.2 ESP 8266	
3.3 Uploaded Data using smart devices in IoT	

### 3.4 Raspberry Pi

#### 3.4.1 Types of Raspberry Pi

#### 3.4.2 Preparing SD card for Pi

### 3.5 Node-RED

#### 3.5.1 History

### 3.6 MQTT

#### 3.6.1 History of MQTT

## **4. PROJECT.....30-44**

### 4.1 Raspberry Pi

4.1.1 Raspberry Pi needs to be setup to be used as a server using various steps as mentioned under this

#### 4.1.2 Installing and setting up Node-RED in Raspberry Pi

### 4.2 ESP8266 module

### 4.3 MQTT module

#### 4.3.1 Publish function

#### 4.3.2 QoS

### 4.4 Hardware

## **5. CONCLUSION.....45**

## **6. REFERENCES .....46**

## LIST OF FIGURES:

### 1. INTRODUCTION

- 1.1. IoT architecture.....11
- 1.2. The four stage architecture of IoT system.....12

### 2. CHAPTER 2

- 2.1 AC to DC Convertor.....14
- 2.2 Water Tank level.....14
- 2.3 Keypad Program.....15
- 2.4 Pin Diagram of 8051.....16
- 2.5 Password Detector using 8051.....17
- 2.6 Pin Diagram of ARM 7.....18
- 2.7 ARM Partnership model.....19

### 3. CHAPTER 3

- 3.1 Arduino UNO board.....20
- 3.2 Arduino IDE.....21
- 3.3 Washing machine implementation.....21
- 3.4 Pin Configuration of ESP8266.....22
- 3.5 Model of ESP8266.....22
- 3.6 Wi-Fi connection established.....23
  - 3.6.1 Uploaded data.....24
  - 3.6.2 Analysis of data.....24
- 3.7 Raspberry Pi.....25
- 3.8 Raspberry Pi 3 model B.....26

3.9 Difference between various models of Raspberry Pi....26

3.10 Working of MQTT.....28

## **4. PROJECT**

### **4.1 Raspberry Pi**

4.1.1 Installing Apache.....29

4.1.2 Apache installed.....30

4.1.3 Node-RED.....33

4.1.4 Node-RED Interface.....34

### **4.2 ESP8266**

4.2.1 MQTT Broker.....35

### **4.3 MQTT**

4.3.1 User Interface.....41

4.3.2 MQTT Flow chart.....42

4.3.3 MQTT Dashboard App.....42

4.4 Hardware .....43

## ABOUT NIELIT



National Institute of Electronics & Information Technology (NIELIT), (erstwhile DOEACC Society), is an Autonomous Scientific Society under the administrative control of Ministry of Electronics & Information Technology (MoE&IT), Government of India, which was set up to carry out Human Resource Development and related activities in the area of Information, Electronics & Communications Technology (IECT). NIELIT is engaged both in Formal & Non-Formal Education in the area of IECT besides development of industry oriented quality education and training programmes in the state of the art areas. NIELIT has endeavoured to establish standards to be the country's premier institution for Examination and Certification in the field of IECT. It is also one of the National Examination Body, which accredits institutes/organizations for conducting courses in IT in the non-formal sector.

At present, NIELIT has thirty five(35) offices located at Agartala, Aizawl, Ajmer, Aurangabad, Calicut, Chandigarh, Chennai, Chuchuyimlang, Churachandpur, Delhi, Gangtok, Gorakhpur, Guwahati, Imphal, Itanagar, Jammu, Jorhat, Kohima, Kolkata, Kokrajhar, Leh, Lucknow, Lunglei, Pasighat, Patna, Ranchi, Ropar (Rupnagar City Centre), Senapati, Shillong, Shimla, Silchar, Srinagar, Srikakulam, Tezpur, Tura with its Head quarters at New Delhi. It is also well networked throughout India with the presence of about 800 institutes.

Over the last two decades, NIELIT has acquired very good expertise in IT training, through its wide repertoire of courses, ranging from 'O' Level (Foundation), 'A' Level (Advance Diploma), 'B' Level (MCA equivalent), 'C' Level (M-Tech level), IT literacy courses such as CCC (Course on Computer Concept), BCC (Basic Computer Course) and other such long term and short term courses in the non-formal sector like courses on Information Security. The basket of activities of NIELIT is further augmented by the wide range of projects that it undertakes. NIELIT has demonstrated its capability and capacity to undertake R&D projects, consultancy services, turnkey projects in office automation, software development, website development etc.



# 1. INTRODUCTION

## 1.1 INTERNET OF THINGS

The Internet of Things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. A thing, in the Internet of Things, can be a person with a heart monitor implant, a farm animal with a biochip transponder, an automobile that has built-in sensors to alert the driver when tire pressure is low or any other natural or man-made object that can be assigned an IP address and provided with the ability to transfer data over a network.

**1.1.1 IoT Definition:** The term Internet of Things generally refers to scenarios where network connectivity and computing capability extends to objects, sensors and everyday items not normally considered computers, allowing these devices to generate, exchange and consume data with minimal human intervention. There is, however, no single, universal definition.

**1.1.2 Enabling Technologies:** The concept of combining computers, sensors, and networks to monitor and control devices has existed for decades. The recent confluence of several technology market trends, however, is bringing the Internet of Things closer to widespread reality. These include Widespread Adoption of IP-based Networking, Computing Economics, Miniaturization, Advances in Data Analytics, and the Rise of Cloud Computing.

**1.1.3 Connectivity Models:** IoT implementations use different technical communications models, each with its own characteristics. Four common communications models described by the Internet Architecture Board include: Device-to-Device, Device-to-Cloud, Device-to-Gateway, and Back-End Data-Sharing. These models highlight the flexibility in the ways that IoT devices can connect and provide value to the user.

**1.1.4 Transformational Potential:** If the projections and trends towards IoT become reality, it may force a shift in thinking about the implications and issues in a world where the most common interaction with the Internet comes from passive engagement with connected objects rather than active engagement with content. The potential realization of this outcome – a “hyper connected world” – is testament to the general-purpose nature of the Internet architecture itself, which does not place inherent limitations on the applications or services that can make use of the technology. Five key IoT issue areas are examined to explore some of the most pressing challenges and questions related to the technology. These include security; privacy; interoperability and standards; legal, regulatory and rights; and emerging economies and development.

**1.1.5 Security:** While security considerations are not new in the context of information technology, the attributes of many IoT implementations present new and unique security challenges. Addressing these challenges and ensuring security in IoT products and services must be a fundamental priority. Users need to trust that IoT devices and related data services are secure from vulnerabilities, especially as this technology becomes more pervasive and integrated into our daily lives. Poorly secured IoT devices and services can serve as potential entry points for cyber attack and expose user data to theft by leaving data streams inadequately protected. The interconnected nature of IoT devices means that every poorly secured device that is connected online potentially affects the security and resilience of the Internet globally. This challenge is amplified by other considerations like the mass-scale deployment of homogenous IoT devices, the ability of some devices to automatically connect to other devices, and the likelihood of fielding these devices in unsecure environments.

As a matter of principle, developers and users of IoT devices and systems have a collective obligation to ensure they do not expose users and the Internet itself to potential harm. Accordingly, a collaborative approach to security will be needed to develop effective and appropriate solutions to IoT security challenges that are well suited to the scale and complexity of the issues.

**1.1.6 Privacy:** The full potential of the Internet of Things depends on strategies that respect individual privacy choices across a broad spectrum of expectations. The data streams and user specificity afforded by IoT devices can unlock incredible and unique value to IoT users, but concerns about privacy and potential harms might hold back full adoption of the Internet of Things. This means that privacy rights and respect for user privacy expectations are integral to ensuring user trust and confidence in the Internet, connected devices, and related services.

Indeed, the Internet of Things is redefining the debate about privacy issues, as many implementations can dramatically change the ways personal data is collected, analyzed, used and protected. For example, IoT amplifies concerns about the potential for increased surveillance and tracking, difficulty in being able to opt out of certain data collection, and the strength of aggregating IoT data streams to paint detailed digital portraits of users. While these are important challenges, they are not insurmountable. In order to realize the opportunities, strategies will need to be developed to respect individual privacy choices across a broad spectrum of expectations, while still fostering innovation in new technology and services

## **1.2 IOT IS DRIVEN BY COMBINATION OF THREE THINGS:**

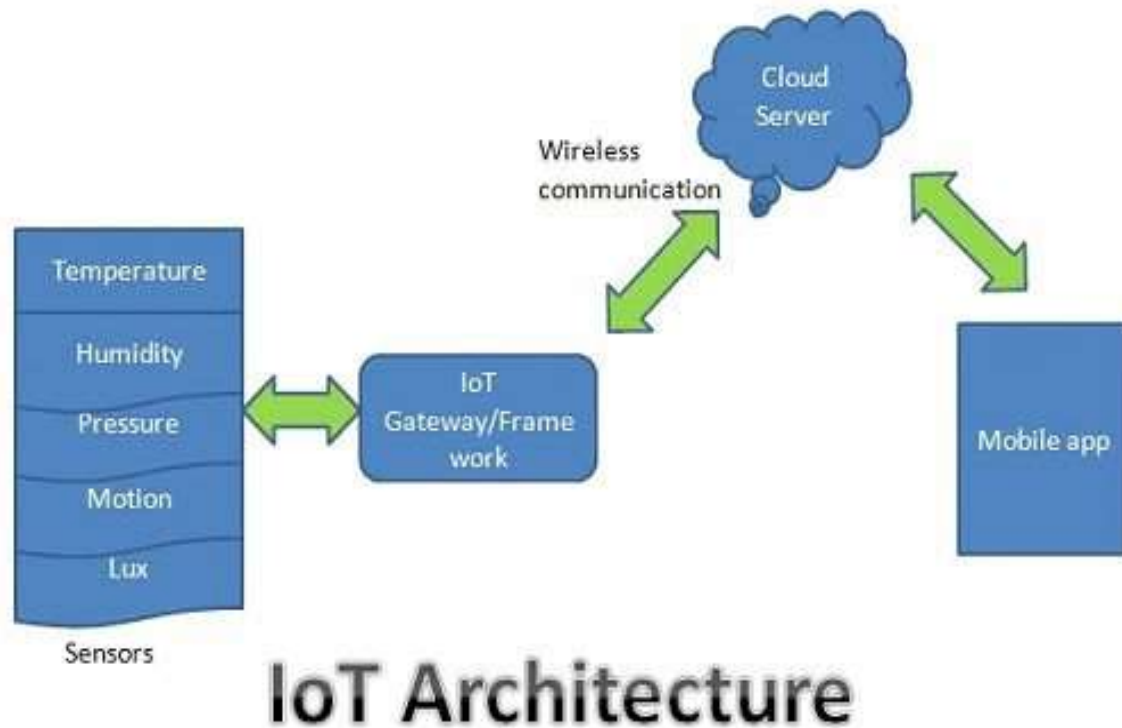
**1.2.1 Sensors and actuators:** The embedded systems can be connected to different sensors for collecting the information. Some of the examples are given below:

- Humidity sensor
- Tilt sensor
- Pressure sensor

- Temperature sensor

They can also be connected to actuators to translate the collected or received information into actions, below are some of the actuators:

- Light emitting diodes (LED)
- Relays



**Figure 1.1:IoT Architecture**

**1.2.2 Connectivity:** The embedded systems can use a range of connectivity to connect with other devices or the internet. Some examples are as below:

- Wi-Fi
- Bluetooth
- 3G

**1.2.3 People and processes:** The information transmitted via the chosen connectivity can be used by people and processes to take action or re-transmit the information to a different embedded system to be used to perform an action using actuators. Below are some examples:

- A gas leak/ smoke is detected by a chemical sensor and the information is transmitted to a monitoring centre. Help is dispatched immediately and the affected are informed.
- A home security system detects intrusion, a call is placed to the authorities and owners alerted. Help dispatched.
- Car can send the diagnostics to the service center and the service center schedules a repair.

**1.2.3.1 Four stage architecture:** Stage 1 of IoT architecture consists of your networked things, typically wireless sensors and actuators. Stage 2 includes sensor data aggregation systems and analog-to-digital data conversion. In Stage 3, edge IT systems perform preprocessing of the data before it moves on to the data center or cloud. Finally, in Stage 4, the data is analyzed, managed, and stored on traditional back end data center systems. Clearly, the sensor/actuator state is the province of operations technology (OT) professionals. So is Stage 2. Stages 3 and 4 are typically controlled by IT.

As a matter of principle, developers and users of IoT devices and systems have a collective obligation to ensure they do not expose users and the Internet itself to potential harm. Accordingly, a collaborative approach to security will be needed to develop effective and appropriate solutions to IoT security challenges that are well suited to the scale and complexity of the issues.

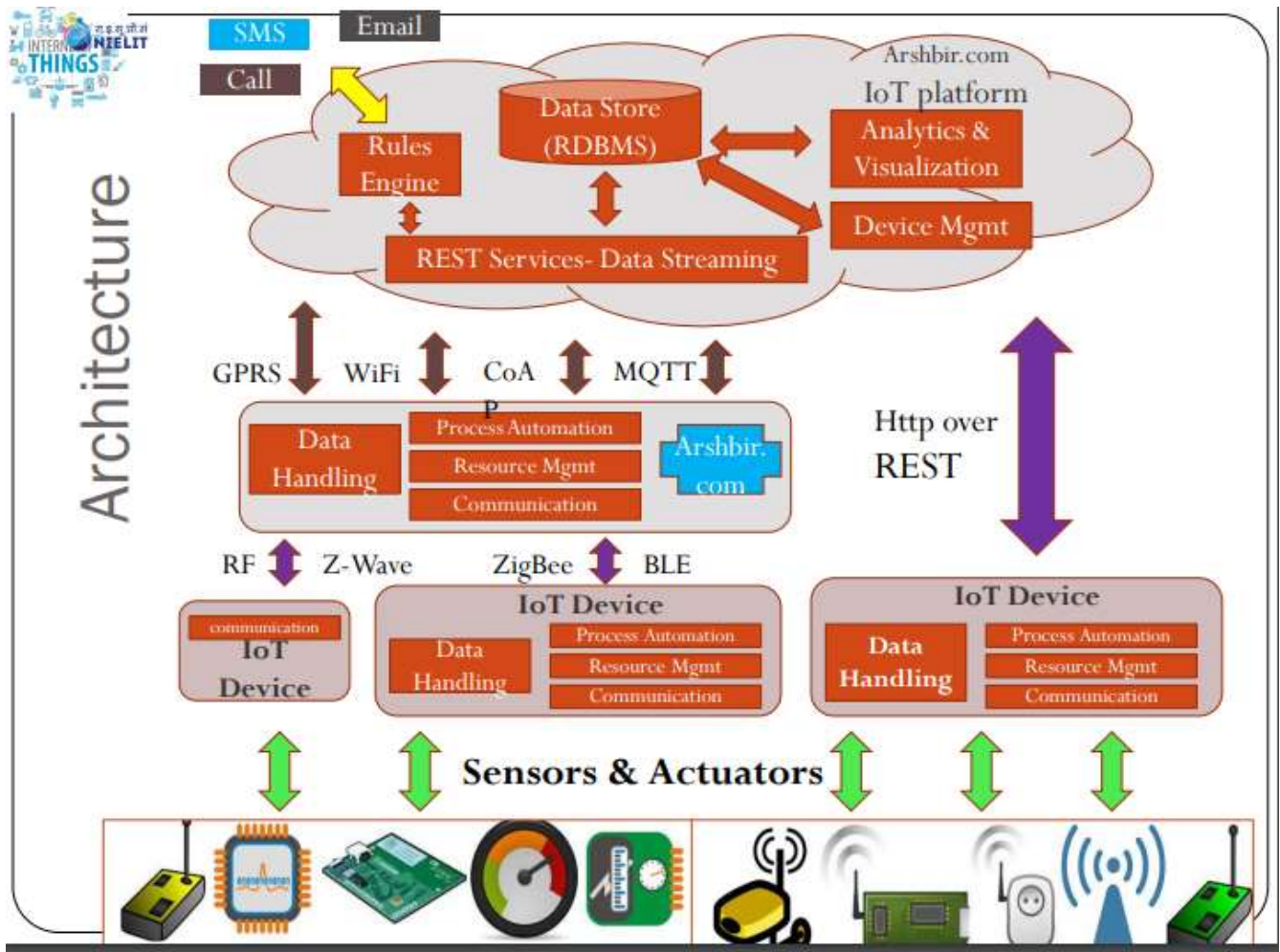


Figure 1.2: The four-stage architecture of an IoT system

## **2.CHAPTER 2: WORK DONE(JANUARY – MARCH)**

### **2.1 Familiarization with C language and Data Structures**

C is a general-purpose, imperative computer programming language, supporting structured programming, lexical variable scope and recursion, while a static type system prevents many unintended operations. By design, C provides constructs that map efficiently to typical machine instructions.

C has now become a widely used for various reasons:

- Easy to learn
- Structured language
- It produces efficient programs
- It can handle low-level activities
- It can be compiled on a variety of computer platforms

A C program basically consists of the following parts –

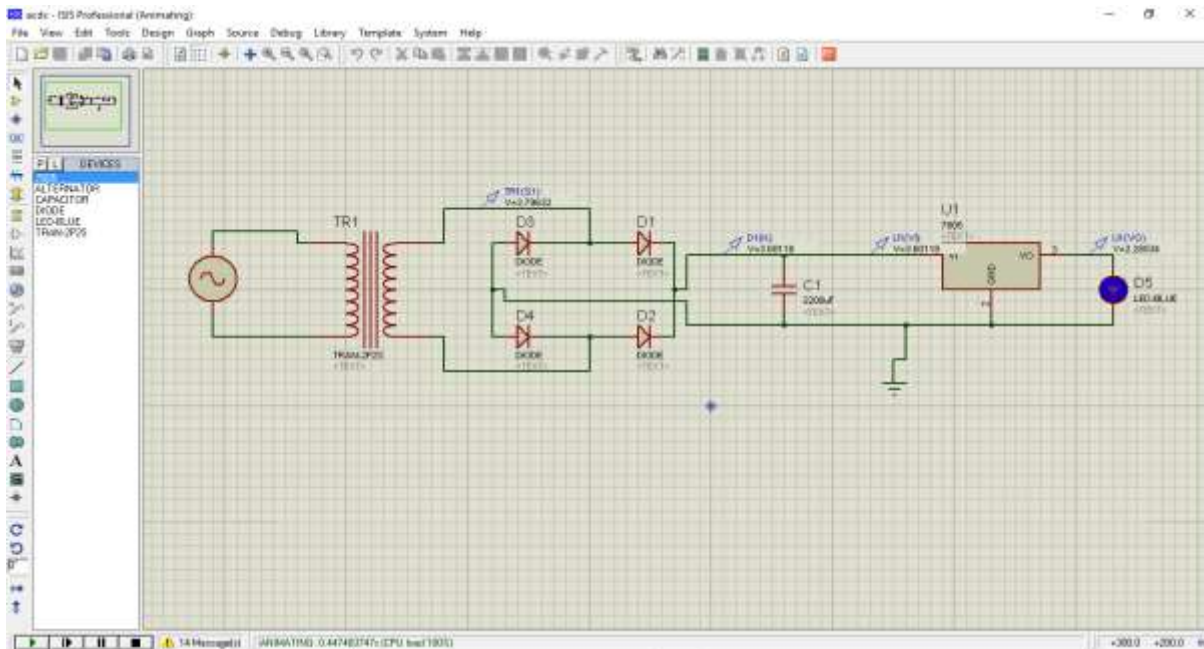
- Preprocessor Commands
- Functions
- Variables
- Statements & Expressions
- Comments

### **2.2 Proteus (Simulation of various electronic circuits)**

Proteus is a great electrical suite for circuit simulation purposes. One can simulate a single processor or multiple ones at the same time. The application is a great alternative for Virtual System Modelling. The micro-controller simulation in Proteus works by applying either a hex file or a debug file to the microcontroller part on the schematic. It is then co-simulated along with any analog and digital electronics connected to it. This enables its use in a broad spectrum of projects.

In this module we implemented the following :

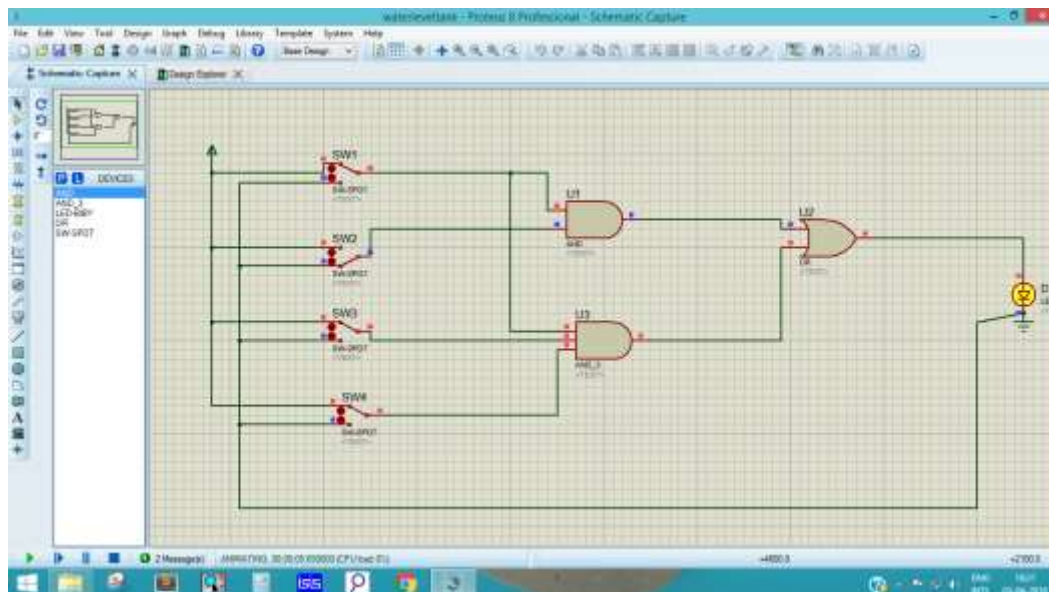
- o AC to DC Converter



**Figure 2.1: AC to DC converter**

The above Proteus design is an AC to DC converter. Firstly the AC source is connected to a step down transformer. A bridge circuit having 4 diodes is connected to the transformer. The output is taken across a parallel Capacitor which is further connected to IC 7805, which is a Voltage Regulator. The output is taken across an LED which glows because of the DC supply.

- Water Tank level implementation



**Figure 2.2 Water Tank level**

In this module we implemented the following:

- ## 2.3 Keil

I, in this module implemented the following:

- 
- ```

008
009
010
011 int i,j;
012 int n,k,m,t,p,q,r,w;
013
014 void delay()
015 {
016   for(i=0;i<5;i++)
017     for(j=0;j<3000;j++);
018 }
019
020 void lcdout(unsigned char a)
021 {
022   rs=0;
023   lcd<4>
024   e=1;
025   delay();
026   e=0;
027 }
028
029 void lcdisplay(unsigned char a)
030 {
031   rs=1;
032   lcd<4>
033   e=1;
034   delay();
035   e=0;
036 }
037
038 void main()
039 {
040   lcd<4>

```

Page | 16

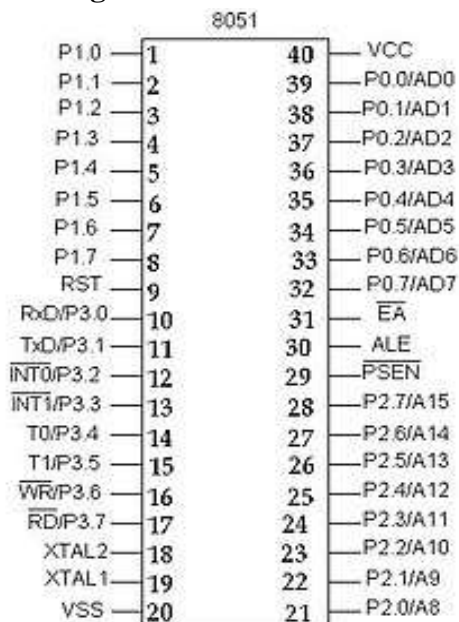


- Password detector
- LED Blinking
- Patterns
- Various Programs implemented through 8051

## 2.4 8051 (Programming & Hardware Simulation)

The Intel 8051 microcontroller is one of the most popular general purpose microcontrollers in use today. The success of the Intel 8051 spawned a number of clones, which are collectively referred to as the MCS-51 family of microcontrollers, which includes chips from vendors such as Atmel, Philips, Infineon and Texas.

### Pin Diagram:



**Figure 2.4:Pin Diagram of 8051**

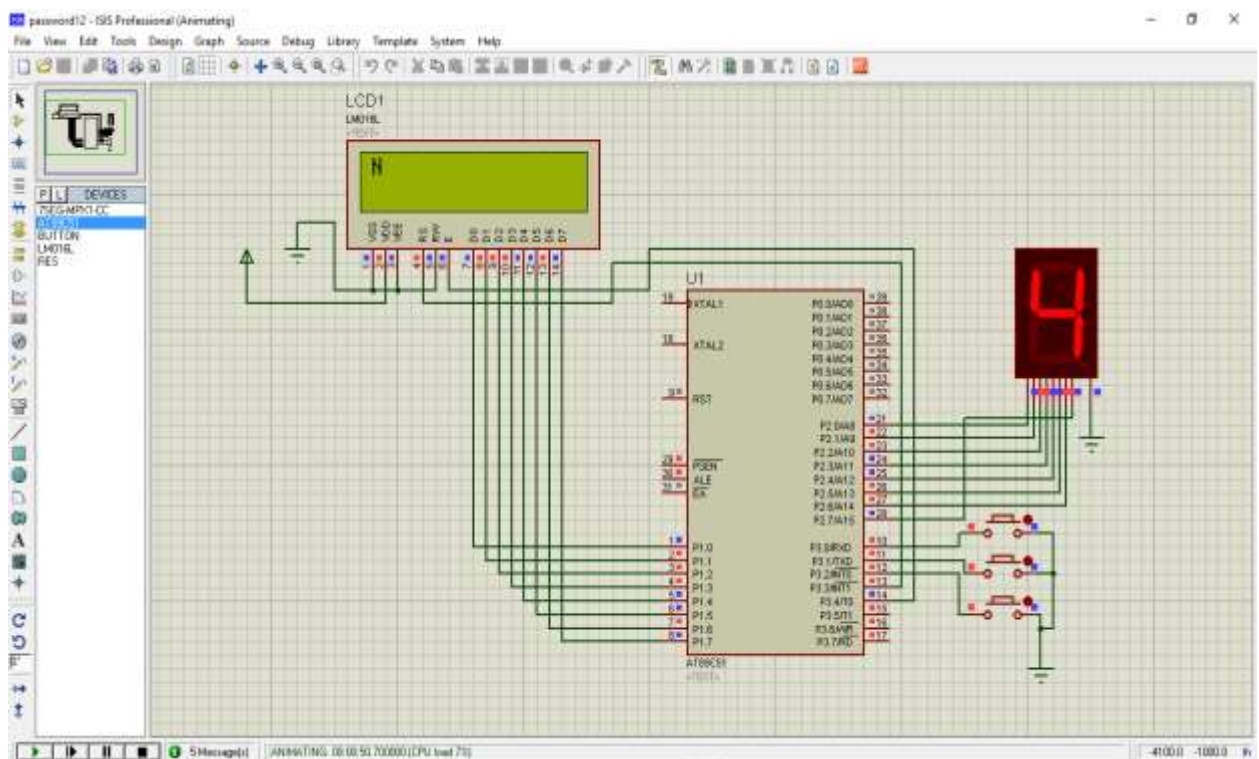
### Features:

- It has 40 pins
- 64K bytes on-chip program memory (ROM)
- 128 bytes on-chip data memory (RAM)

- Four register banks
- 8-bit bidirectional data bus
- 16-bit unidirectional address bus
- 32 general purpose registers each of 8-bit
- 16 bit Timers (usually 2)
- Three internal and two external Interrupts
- Four 8-bit ports
- 16-bit program counter

I, in this module implemented the following:

- Password Detector using 8051



**Figure 2.5: Password Detector using 8051**

The above circuit uses 3 buttons, one for increasing the number, second for decrementing it, third to input the number. If the entered number matches the already input code, the display will show Y for yes else N for No, i.e., the entered password is incorrect.

- Patterns using LEDs
- LCD Interfacing
- Buzzer program
- 7 Segment LED programs, among others

## 2.5 ARM 7

ARM stands for Advanced RISC Machines. ARM-Advanced RISC Machine is a 32-bit RISC(Reduced Instruction Set Computer) processor architecture developed by ARM Holdings. We started with ARM7 TDMI based NXP controller LPC2148. LPC2148 is manufactured by NXP Semiconductor(Phillips) and it is preloaded with many in-built features and peripherals.

### Pin Diagram:

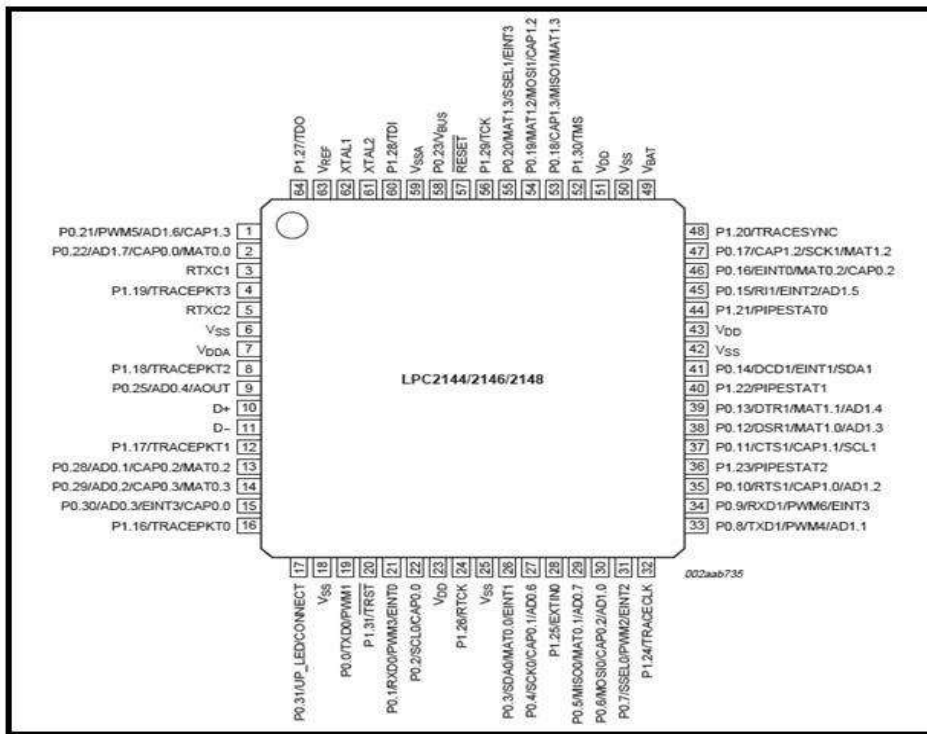


Figure 2.6: Pin Diagram of ARM 7

### Features:

- 64 I/O pins
- Crystal for LPC2148: 12Mhz
- Operating Supply: 3.3V
- 40 kB of on-chip static RAM
- 512 kB of on-chip flash memory
- 5 GPIO are present namely
  - PINSEL- To choose the configuration of the pin(one pin can do upto 4 functions )
  - IOPIN-read or write values
  - IODIR-used to set the direction i.e.either input or output of individual pins(0-input,1-output)
  - IOSET-set value high
  - IOCLR-set value low



### Figure 2.7:ARM Partnership model

### 3. CHAPTER 3: WORK DONE (APRIL-MAY)

#### 3.1 ARDUINO UNO

Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on wiring), and the arduino software (IDE), based on processing.

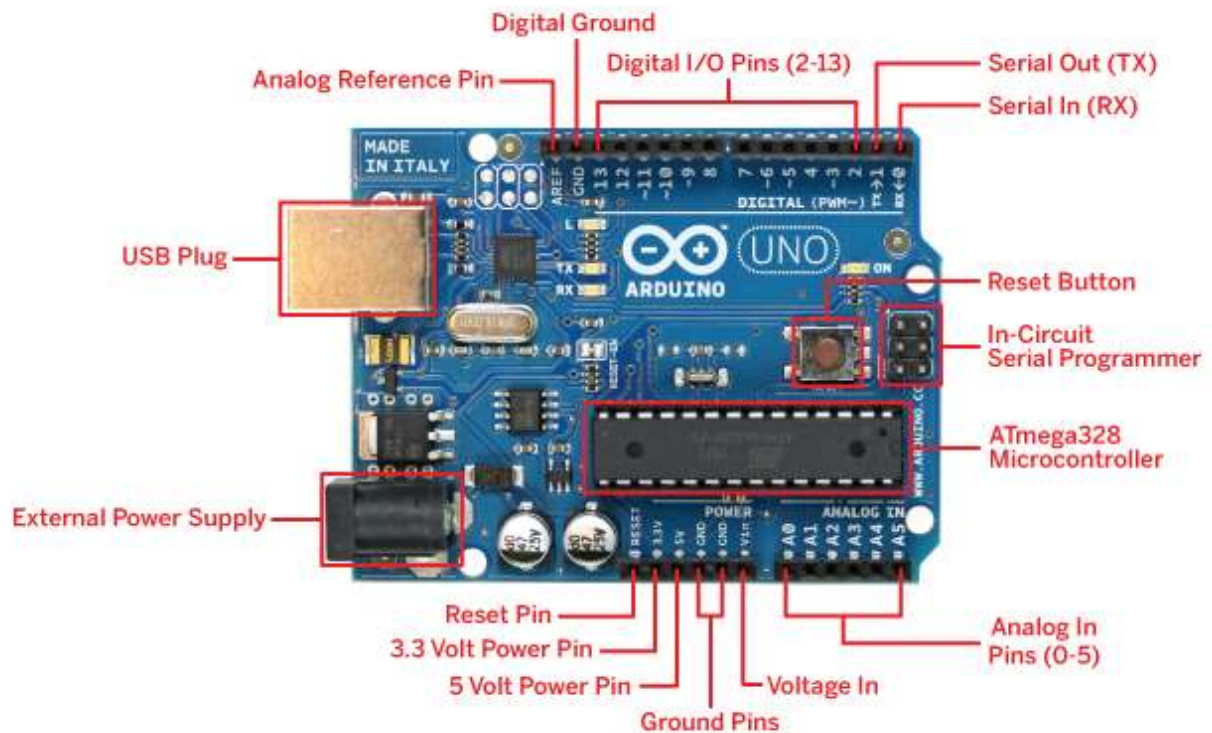


Figure 3.1: Arduino UNO Board

#### FEATURES:

- Processor: 16 Mhz ATmega328
- Flash memory: 32 KB
- RAM: 2kb
- Operating Voltage: 5V

- Input Voltage: 7-12 V
- Number of analog inputs: 6
- Number of digital I/O: 14 (6 of them pwm)

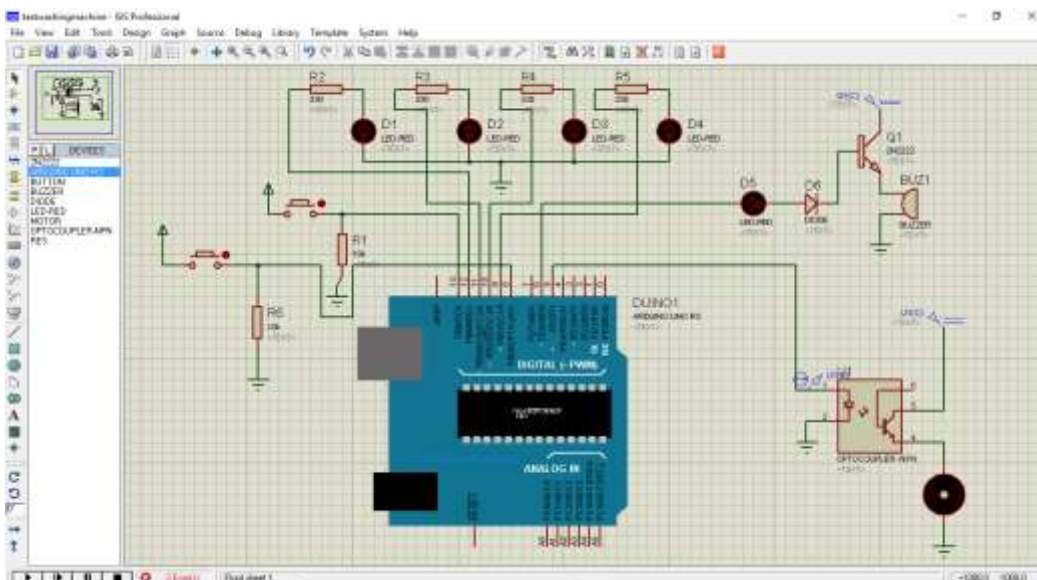
### ARDUINO IDE:



**Figure 3.2: Arduino IDE**

I, in this module implemented the following :

- Washing Machine Implementation

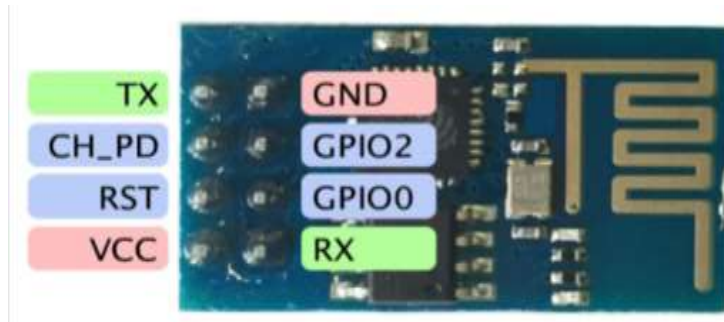


**Figure 3.3: Washing Machine Implementation**

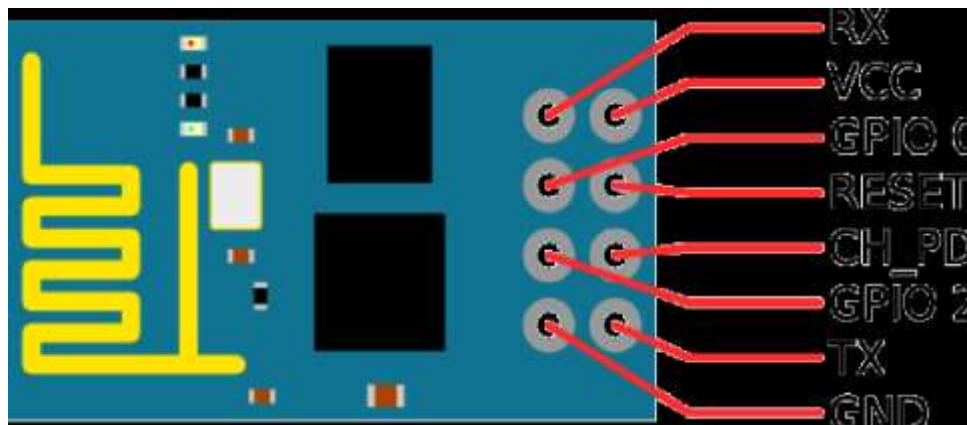


- LED patterns
- Using Motor, Relays, etc

### 3.2 ESP 8266



**Figure 3.4: Pin configuration of ESP 8266**



**Figure 3.5: Module of ESP8266**

ESP8266 offers a complete and self-contained Wi-Fi networking solution, allowing it to either host the application or to offload all Wi-Fi networking functions from another application processor.

ESP8266 is an impressive, low cost Wi-Fi module suitable for adding Wi-Fi functionality to an existing microcontroller project via a UART serial connection.

The module can even be reprogrammed to act as a standalone Wi-Fi connected device—just add power!

**The feature list is impressive and includes:**

- CPU ( 32 bit, 26MHz-52MHz, 64KB instruction RAM, 64KB boot ROM, 96KB data RAM),

- CPU clock speed can reach maximum value of 160 MHz.
- 802.11 b/g/n protocol
- Wi-Fi Direct (P2P)
- Integrated TCP/IP protocol stack
- It requires 3.3V power
- Real Time Operation System (RTOS) is enabled

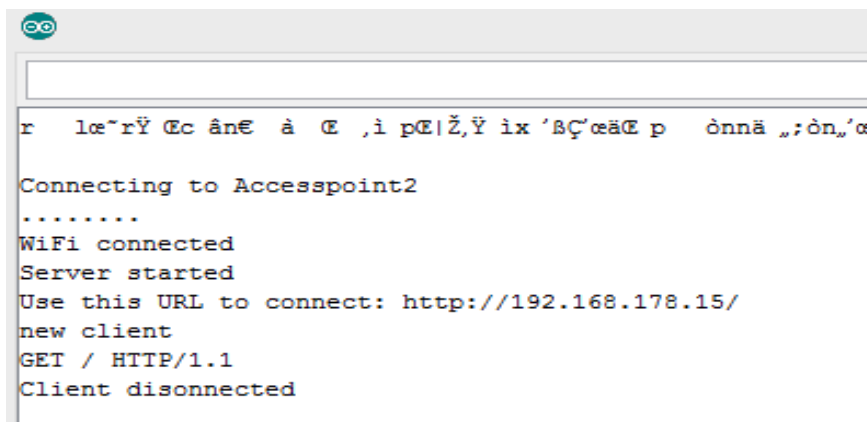
ESP8266EX is among the most integrated WiFi chips in the industry with the size of 5mm x 5mm ; it integrates the antenna switches, RF balun, power amplifier, low noise receive amplifier, filters, power management modules while requires minimal external circuitry. The entire solution, including front-end module, is designed to occupy minimal PCB area.

ESP8266EX has been designed for mobile, wearable electronics and Internet of Things applications with the aim of achieving the lowest power consumption with a combination of several proprietary technologies. The power saving architecture operates in 3 modes: active mode, sleep mode and deep sleep mode.

### 3.3 Smart devices using IoT

I, in this module implemented the following:

- Sending data from sensors like smoke detector, humidity sensor, etc on “Arshbir.com”
- Analysing the sent data over the internet using graphs



```

r  lœ~rŸ @c ân€ à @ ,i p@|Ž,Ÿ ix 'BÇ'œä@ p  ònnä „;òn„'œ

Connecting to Accesspoint2
.....
WiFi connected
Server started
Use this URL to connect: http://192.168.178.15/
new client
GET / HTTP/1.1
Client disconnected

```

**Figure3.6: WIFI connection established**





**Figure 3.6.1: Uploaded Data**



**Figure 3.6.2: Analysis of Data**

### 3.4 Raspberry Pi

A Raspberry Pi is a credit card-sized computer originally designed for education, inspired by 1981 BBC Micro. Creator Eben Upton's goal was to create a low-cost device that would improve programming skills and hardware understanding at pre- university level. But thanks to its small size and accessible price, it was quickly adopted by thinkers, makers and enthusiasts for projects that require more than a basic microcontroller (such as Arduino devices).

The Raspberry Pi is slower than a modern laptop or desktop but is still a complete Linux Computer and can provide all the expected abilities that implies, at a low power consumption level.



**Figure 3.7: Raspberry Pi**

### **3.4.1 Different Types of Raspberry Pi Models**

The different types of raspberry pi models are following

- Raspberry Pi 1 model B
- Raspberry Pi 1 model A
- Raspberry Pi 1 model A+
- Raspberry Pi Zero
- Raspberry Pi 2

#### **Features:**

- quad-core Cortex-A53 CPU at 1.2GHz
- with a VideoCore IV GPU clocked at 300-400MHz (3D clocked at 300MHz, video at 400MHz)
- 1GB of RAM,
- 802.11n wireless
- power consumption of 4W.

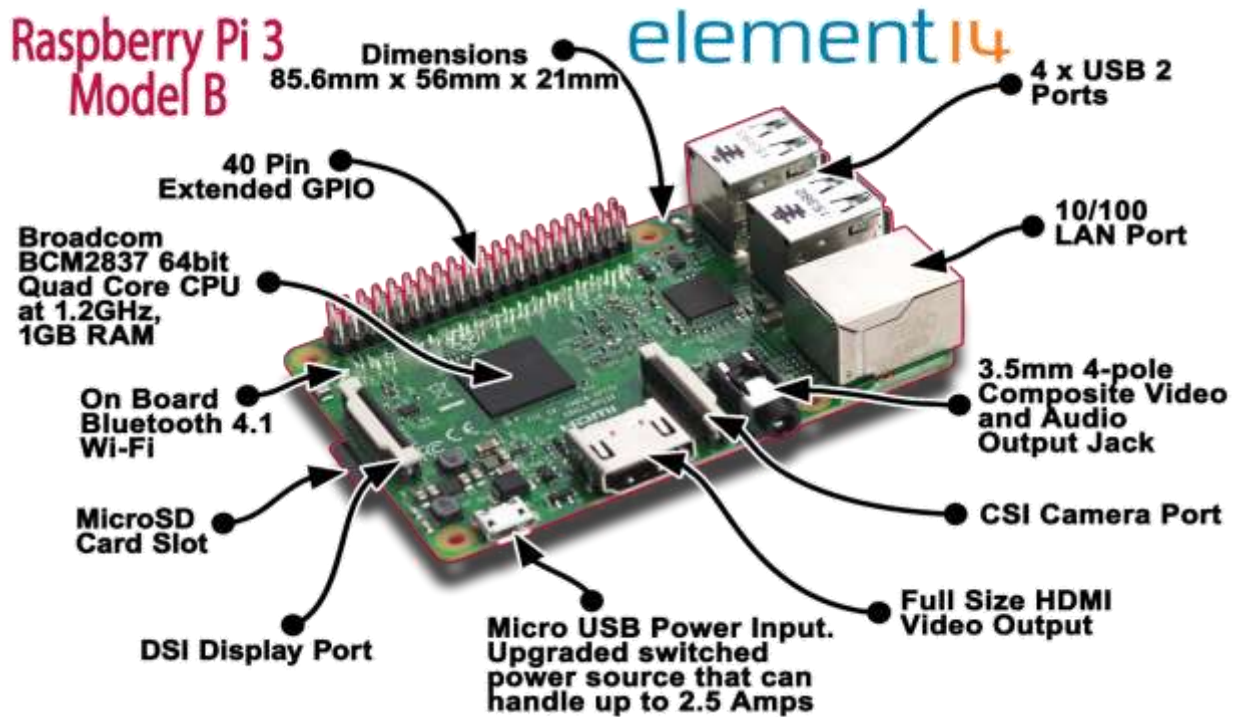


Figure 3.8: Raspberry pi 3 Model B

|                               | Raspberry Pi 3 Model B                                                                     | Raspberry Pi 2 Model B                                                                     | Raspberry Pi Model B+                                                | Raspberry Pi Model A+                                                |
|-------------------------------|--------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|----------------------------------------------------------------------|----------------------------------------------------------------------|
| <b>Processor Chipset</b>      | Broadcom BCM2837 64Bit Quad Core Processor powered Single Board Computer running at 1.2GHz | Broadcom BCM2836 32Bit Quad Core Processor powered Single Board Computer running at 900MHz | Broadcom BCM2835 32Bit SoC full HD multimedia applications processor | Broadcom BCM2835 32Bit SoC full HD multimedia applications processor |
| <b>GPU</b>                    | Videocore IV                                                                               | Videocore IV                                                                               | Videocore IV                                                         | Videocore IV                                                         |
| <b>Processor Speed</b>        | QUAD Core @1.2 GHz                                                                         | QUAD Core @900 MHz                                                                         | Single Core @700 MHz                                                 | Single Core @700 MHz                                                 |
| <b>RAM</b>                    | 1GB SDRAM @ 400 MHz                                                                        | 1GB SDRAM @ 400 MHz                                                                        | 512 MB SDRAM @ 400 MHz                                               | 256 MB SDRAM @ 400 MHz                                               |
| <b>Storage</b>                | MicroSD                                                                                    | MicroSD                                                                                    | MicroSD                                                              | MicroSD                                                              |
| <b>USB 2.0</b>                | 4x USB Ports                                                                               | 4x USB Ports                                                                               | 4x USB Ports                                                         | 1x USB Port                                                          |
| <b>Max Power Draw/voltage</b> | 2.5A @ 5V                                                                                  | 1.8A @ 5V                                                                                  | 1.8A @ 5V                                                            | 1.8A @ 5V                                                            |
| <b>GPIO</b>                   | 40 pin                                                                                     | 40 pin                                                                                     | 40 pin                                                               | 40 pin                                                               |
| <b>Ethernet Port</b>          | Yes                                                                                        | Yes                                                                                        | Yes                                                                  | No                                                                   |
| <b>WiFi</b>                   | Built in                                                                                   | No                                                                                         | No                                                                   | No                                                                   |
| <b>Bluetooth LE</b>           | Built in                                                                                   | No                                                                                         | No                                                                   | No                                                                   |

Figure 3.9: Difference between various models of raspberry pi

### 3.4.2 Preparing SD card for pi

- Download NOOBS(New Out Of Box Software) from the raspberrypi.org downloads page
- Insert a (4 GB+) SD Card into your computer
- Format the disk (Windows)
- Extract the file you downloaded in Step 1
- Copy the files you just extracted to your SD Card

Irrespective of its size, Raspberry Pi is a powerhouse of a computer. It can drive HDMI displays, mouse, keyboard, camera – above all it runs full featured Linux distribution. • Not only computer it is hardware prototyping tool. • The Pi has bi-directional I/O pins, which can be used to drive LEDs, spin motors, or read button presses.

## 3.5 Node-RED

Node-RED is a flow-based development tool developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things. Node-RED provides a browser-based flow editor, which can be used to create JavaScript functions.

### 3.5.1 History of Node-RED

- Flow-based Programming
- Invented by J. Paul Morrison in the 1970s, flowbased programming is a way of describing an application's behaviour as a network of black-boxes, or “nodes” as they are called in Node-RED.
- Each node has a well-defined purpose; it is given some data, it does something with that data and then it passes that data on.
- The network is responsible for the flow of data between the nodes.

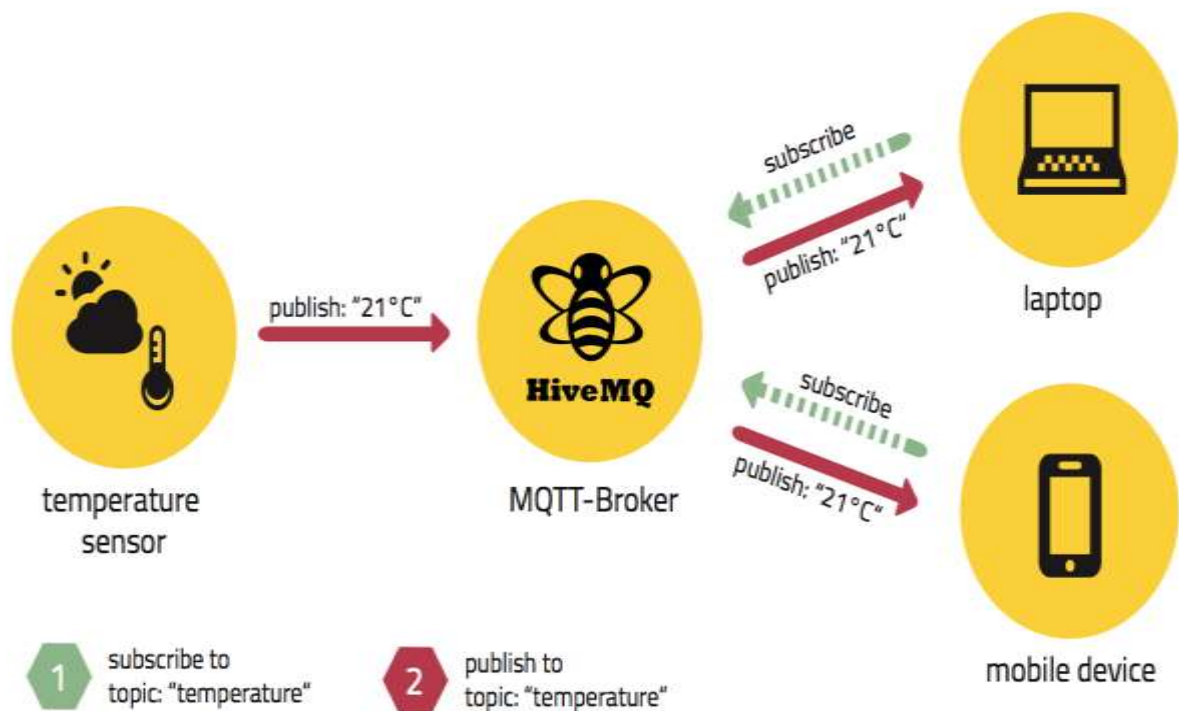
## 3.6 MQTT

MQTT is a binary client-server publish/subscribe messaging transport protocol It is lightweight, open, simple, and easy to implement. Designed with a minimal protocol overhead, This protocol is a good choice for a variety of (M2M) and IoT applications MQTT utilizes many characteristics of the TCP transport. So the minimum requirement for using MQTT is a working TCP stack, which is now

available for even the smallest microcontrollers. The most recent version of MQTT is 3.1.1, which has many improvements over the first public MQTT release, MQTT 3.1.

### 3.6.1 History of MQTT

- MQTT was developed by Andy Stanford-Clark (IBM) and Arlen Nipper (Eurotech; now Cirrus Link) in 1999 for the monitoring of an oil pipeline through the desert.
- The goals were to have a protocol, which is bandwidth-efficient and uses little battery power, because the devices were connected via satellite link and this was extremely expensive at that time.



**Figure 3.10: Working of MQTT**

## 4. PROJECT

### Title: Home Automation using Raspberry Pi as a server

Modules:

4.5 Raspberry Pi

4.6 ESP8266

4.7 MQTT

4.8 Hardware

### 4.1 Module 1

4.1.1 Raspberry Pi needs to be setup to be used as a server in this project using various steps.

The steps are:

**4.1.1.1 First requirement is to have a Raspberry Pi computer connected to the internet.**

**4.1.1.2 Set up an Apache web server**

Apache is a popular web server application you can install on the Raspberry Pi to allow it to serve web pages. On its own, Apache can serve HTML files over HTTP. With additional modules it can serve dynamic web pages using scripting languages such as PHP.

- **Install Apache** Open a terminal window by selecting Accessories > Terminal from the menu.
- Install the apache2 package by typing the following command into the terminal and pressing Enter:  
`sudo apt-get install apache2 -y`

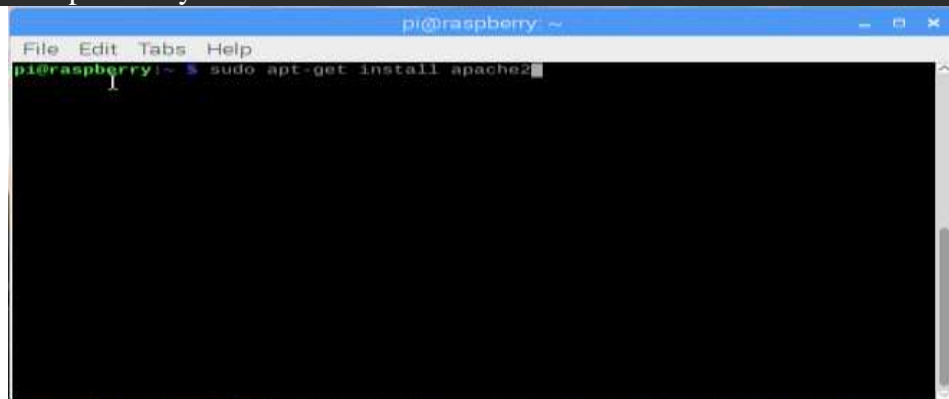


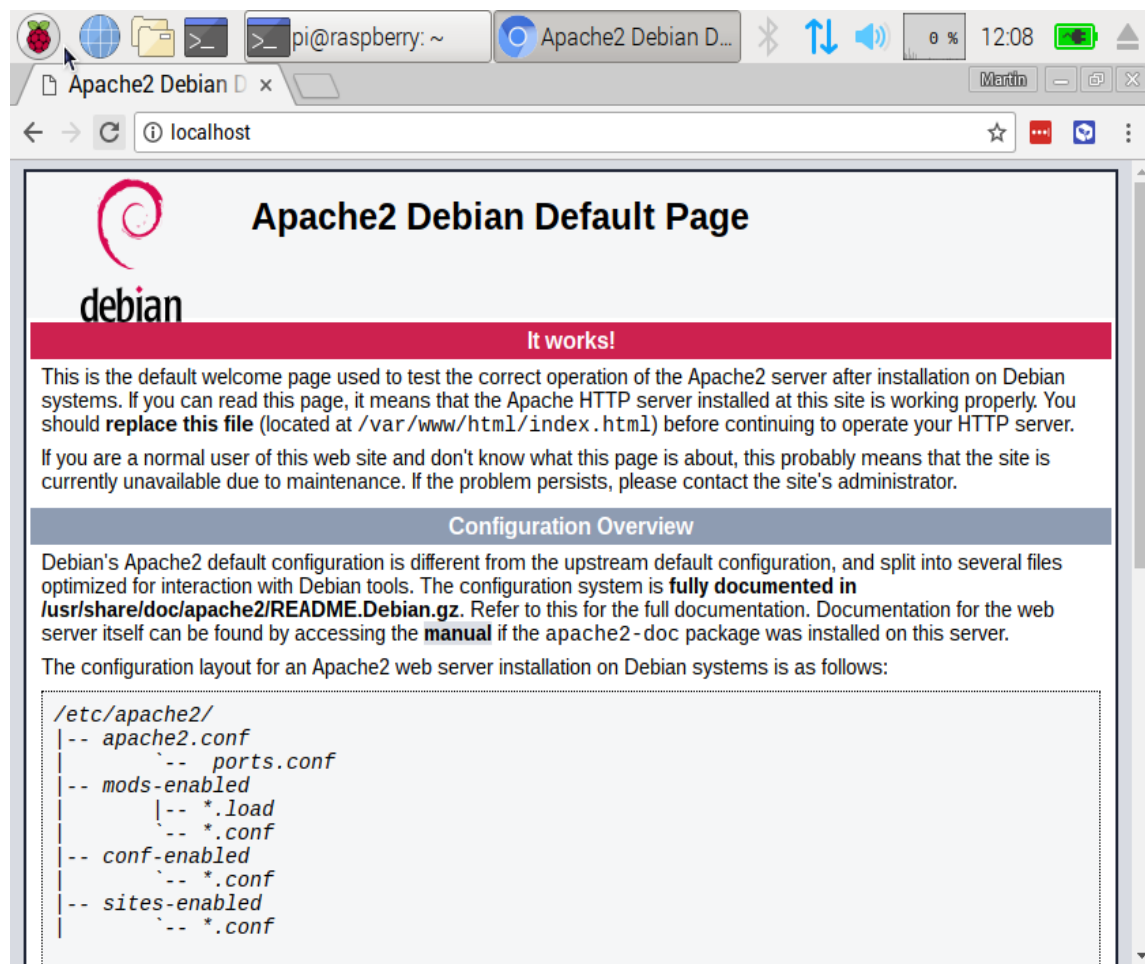
Figure 4.1.1: Installing apache

- **Test the web server**

By default, Apache puts a test HTML file in the web folder that you will be able to view from your Pi or another computer on your network.

Open the Apache default web page on your Raspberry Pi:

- Open Chromium by selecting Internet > Chromium Web Browser from the menu.
- Enter the address `http://localhost`.  
You should see this in your browser window:



**Figure 4.1.2: Apache installed**

This means you have Apache working!

You will also be able to open this web page from any other computer on your network using the IP address of your Raspberry Pi, e.g. `http://192.168.1.10`.

To find out your Raspberry Pi's IP address, type `hostname -I` into the terminal window. Your Raspberry Pi's IP address is a really useful and will allow you to remotely access it. In this project, this



IP address is the one that helps connect the MQTT Dashboard to the project and control the operation of appliances.

- **Changing the default web page**

This default web page is just a HTML file on the file system. It is located at `/var/www/html/index.html`.

- Navigate to this directory in the terminal and have a look at what's inside:

```
cd /var/www/html
ls -al
```

You should see this in the window:

```
total 12
drwxr-xr-x  2 root root 4096 Jan  8 01:29 .
drwxr-xr-x  3 root root 4096 Jan  8 01:28 ..
-rw-r--r--  1 root root  177 Jan  8 01:29 index.html
```

This shows that there is one file in `/var/www/html/` called `index.html`. `.` refers to the directory itself `/var/www/html`, and `..` refers to the parent directory `/var/www/`.

What the columns mean

1. The permissions of the file or directory
2. The number of files in the directory (or 1 if it's a file).
3. The user that owns the file or directory
4. The group that owns the file or directory
5. The size of the file or directory
6. The date and time of the last modification

As you can see, the `html` directory and `index.html` file are both owned by the `root` user, so you'll need to use `sudo` to edit them.

You can edit this file using `leafpad`:

```
sudo leafpad index.html
```

If you make a change to the file, save it, and refresh the browser, you will see your change appear.

#### **4.1.1.3 Install PHP**

PHP is a pre-processor: its code that runs when the server receives a request for a web page via a web browser. It works out what needs to be shown on the page, and then sends that page to the browser. Unlike static HTML, PHP can show different content under different circumstances. Other languages



are also capable of doing this, but since WordPress is written in PHP, that's what we need to use this time. PHP is a very popular language on the web: huge projects like Facebook and Wikipedia are written in PHP.

- Install the PHP and Apache packages with the following command:

```
sudo apt-get install php -y
```

- **Test PHP**
- Create the file index.php:

```
sudo leafpad index.php
```

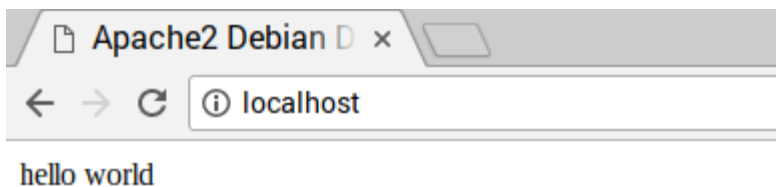
- Put some PHP content in it:

```
<?php echo "hello world"; ?>
```

- Save the file.
- Delete index.html, because it takes precedence over index.php:  

```
sudo rm index.html
```

  
Refresh your browser. You should see “hello world”. This page is not dynamic, but it is still served by PHP.



If you see the raw PHP above instead of “hello world”, reload and restart Apache like so:

```
sudo service apache2 restart
```

- Edit index.php to include some dynamic content, for example:

```
<?php echo date('Y-m-d H:i:s'); ?>
```

Or show your PHP info:

```
<?php phpinfo(); ?>
```

#### 4.1.1.4 Install MySQL

MySQL (pronounced *My Sequel* or *My S-Q-L*) is a popular database engine. Like PHP, it's widely used on web servers, which is why projects like WordPress use it, and why those projects are so popular.

Install the MySQL Server and PHP-MYSQL packages by entering the following command into the terminal window:

```
sudo apt-get install mysql-server php-mysql -y
```

Now restart Apache:

```
sudo service apache2 restart
```

Thus, we have installed apache, php and mysql to make pi a local server.

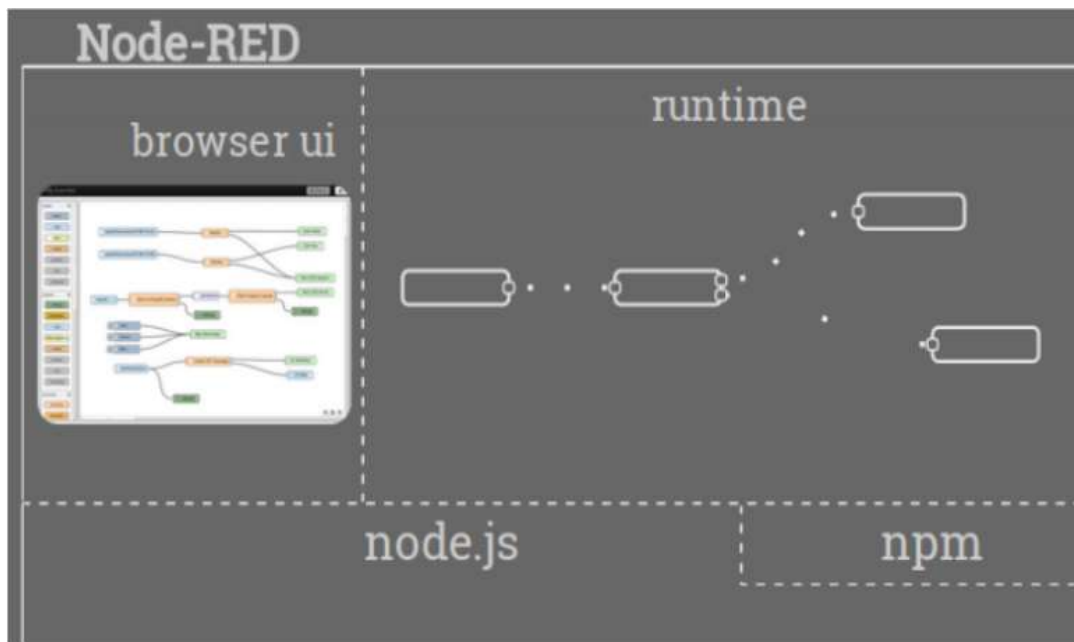
This is a step by step process where the Raspberry Pi has been setup as a server.

#### 4.1.2 Installing and setting up Node-RED in Raspberry Pi

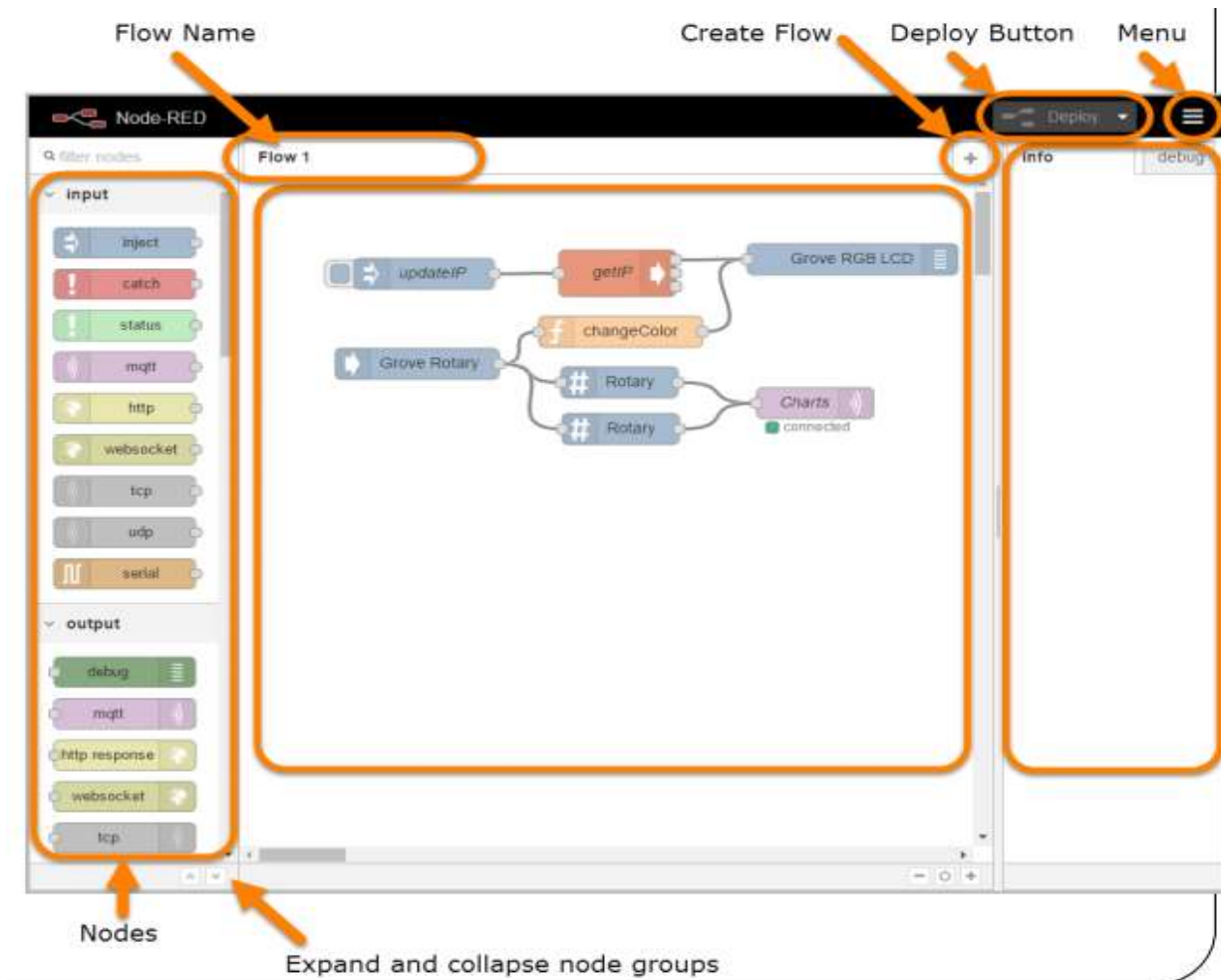
Node-RED is a flow based development tool. Each node has a well-defined purpose; it is given some data, it does something with that data and then it passes that data on. The network is responsible for the flow of data between the nodes.

##### 4.1.2.1 Architecture and examples

1. It is Node.js v8-engine driven so it's fast and it is all about the events.
2. It has a Single-threaded event queue and built for fairness.
3. It uses JavaScript front and back, thus has only one language runtime to deal with.



**Figure 4.1.3: Node-RED**



**Figure 4.1.4: Node-RED interface**

There are three main types of nodes:

1. Input Nodes (e.g. inject)
2. Output Nodes (e.g. debug)
3. Processing Nodes (e.g. function)

#### 4.1.2.3 Installing Node-Red-Dashboard

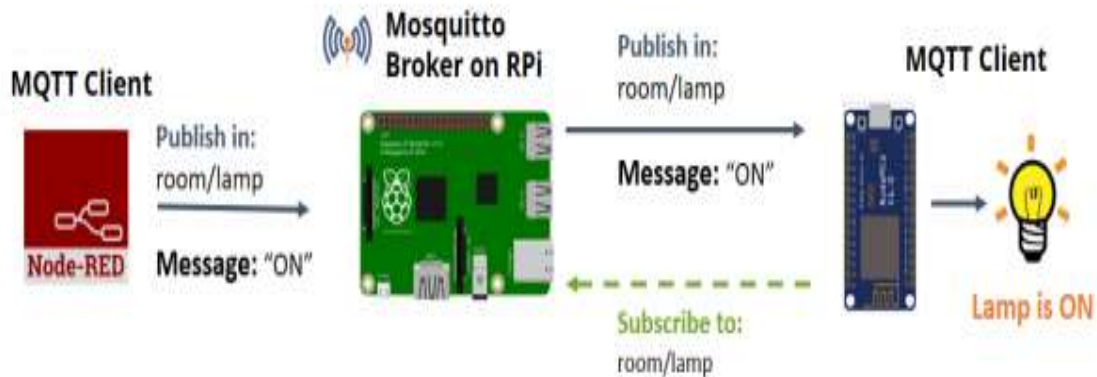
```
$ sudo apt-get install npm
```

```
$ sudo npm install node-red-dashboard
```

```
$ sudo reboot
```

#### 4.1.2.4 ESP8266 and Node-RED with MQTT (Publish and Subscribe)

- a) Installing Mosquitto Broker
- b) Establishing an MQTT communication with Node-RED
- c) Preparing your Arduino IDE



**Figure 4.2: MQTT Broker**

##### **a) Installing Mosquitto Broker**

In MQTT, the broker is primarily responsible for receiving all messages, filtering the messages, decide who is interested in it and then publishing the message to all subscribed clients.

Mosquitto Broker to be installed on Raspberry Pi:

```
$ sudo apt install mosquitto
```

```
$ sudo systemctl enable mosquitto.service
```

(To see if Mosquitto broker was successfully installed)\$ mosquitto-v

##### **b) Establishing an MQTT communication with Node-RED**

- Dashboard Layout- On the top right corner of the Node-RED window, select the Layout tab under the dashboard tab. Create a tab called Room and inside the Room tab, create a group: Lamp

- Creating Flow switch – this will control the ESP8266 output
- Mqtt output node – this will publish a message to the ESP8266 accordingly to the switch state
- Click the Add newmqtt-broker option.
- Type localhost/IP address of server in the server field
- All the other settings are configured properly by default.
- Press Add and the MQTT output node automatically connects to your broker.
- Switch – the switch sends an on string message when it's on; and sends an off string message when it's off. This node will publish on the room/lamp topic.ESP will subscribe to this topic, to receive its messages.
- Mqtt output node. This node is connected to the mosquitto broker and it will publish in the room/lamp topic.
- Your Node-RED flow is ready. Click the Deploy button on the top right corner. The Node-RED application is ready. To see how your dashboard looks go to <http://your-pi-ipaddress/ui>.

### c) Preparing your Arduino IDE code

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
void setup_wifi();
void callback(char* topic, byte* payload, unsigned int length);
void reconnect();

const char* ssid = "demo";
const char* password = "12345678";

const char* mqtt_server = "192.168.43.56";
int gpio2_pin = 2;

WiFiClient espClient;
PubSubClient client(espClient);
//////////
const int lamp = 2; //4;
const int buttonPin=0 ; // gpio0
```

```

int buttonState, buttonValue;
long lastDebounceTime = 0; // the last time the output pin was toggled
long debounceDelay = 350; // the debounce time; increase if the output flicks 50
boolean isWifiConnected = false;

//////////
void setup() {
    pinMode(gpio2_pin, OUTPUT);
    pinMode(buttonPin, INPUT);
    // digitalWrite(gpio2_pin, HIGH);
    // delay(1000);
    // digitalWrite(gpio2_pin, LOW);

    Serial.begin(115200);
    setup_wifi();
    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);
    reconnect();
}

void setup_wifi(){

    delay(10);
    // We start by connecting to a WiFi network
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);

    if (WiFi.status() != WL_CONNECTED) {
        Serial.println("wifi not connected ... exiting setup_wifi ");
        isWifiConnected = false;
        delay(500);
    // Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
    isWifiConnected = true;
}

void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print(" [");
    Serial.print(topic);

```

```

Serial.print("] ");
String messageTemp="";
for (int i = 0; i < length; i++) {
  Serial.print((char)payload[i]);
  messageTemp += (char)payload[i];
}
// Feel free to add more if statements to control more GPIOs with MQTT
// If a message is received on the topic room/lamp, you check if the message is either on or off.
Turns the lamp GPIO according to the message
If (topic=="room/lamp")
{
  Serial.print("Changing Room lamp to ");
  If (messageTemp == "on"){
    digitalWrite(lamp, HIGH);
    Serial.print("on");
  }
else if(messageTemp == "off"){
  digitalWrite(lamp, LOW);
  Serial.print("off");
}
}
Serial.println();
// if((char)payload[0] == 'o' && (char)payload[1] == 'n') { //on
//   digitalWrite(gpio2_pin, HIGH);
//   Serial.print("pin high");
// }
// else if((char)payload[0] == 'o' && (char)payload[1] == 'f' && (char)payload[2] == 'f') //off
// { digitalWrite(gpio2_pin, LOW);
//   Serial.print("pin low");
// }
// Serial.println();
}

void reconnect() {
  // check for wifi connection if not connected call setup_wifi
  if (isWifiConnected == false) setup_wifi();

  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP8266Client")) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      if ( buttonValue ==0)
        if (client.publish("room/lamp", "off") ) {

```

```

        Serial.println("Publish off success in reconnect");
    }else {
        Serial.println("Publish off failed in reconnect");
    }
    client.publish("room/lamp", "off");
    if ( buttonValue ==1)
        if (client.publish("room/lamp", "on") ) {
            Serial.println("Publish on success in reconnect");
        }else {
            Serial.println("Publish on failed in reconnect");
        }
    // ... and resubscribe
    client.subscribe("room/lamp");
} else {
    Serial.print("failed, rc=");
    Serial.print(client.state());
    Serial.println(" try again in 5 seconds");
    // Wait 5 seconds before retrying
    delay(2000);
}
}
}

void loop() {

    // check for wifi connection if not connected call setup_wifi
    if (isWifiConnected == false) setup_wifi();

    if (isWifiConnected == true)
    { if (!client.connected()) {
        reconnect();
    }
    if(!client.loop())
        client.connect("ESP8266Client");
    }
    //////////// button
    buttonState = digitalRead(buttonPin);

    //filter out any noise by setting a time buffer
    if ( ( millis() - lastDebounceTime) > debounceDelay) {

        if (buttonState == LOW )
        {
            while ((buttonState = digitalRead(buttonPin)) ==LOW);
            if(buttonValue==1) buttonValue =0;
            else buttonValue =1;
        }
    }
}

```



```

if ( buttonValue == 1)
{ digitalWrite(lamp, HIGH);
  Serial.println("LED High");
  lastDebounceTime = millis();
  if ( isWifiConnected && client.connect("ESP8266Client"))
  {
    if (client.publish("room/lamp", "on") ) {
      Serial.println("Publish on success");
    }else {
      Serial.println("Publish on failed");
    }
  }
  //delay(100);
}
if ( buttonValue ==0)
{ digitalWrite(lamp, LOW);
  Serial.println("LED low");
  lastDebounceTime = millis();
  if (isWifiConnected &&client.connect("ESP8266Client"))
  {
    if (client.publish("room/lamp", "off") ) {
      Serial.println("Publish off success");
    }else {
      Serial.println("Publish off failed");
    }
  }
}
}
}
}
}

```

## 4.2 Module 2

ESP8266 has its function in this project just to provide connection amongst modules through Wi-Fi hotspot. A hotspot is created to which ESP connects and makes it possible to operate the hardware through connection with raspberry server and thus mqtt in Node-RED. The code mentioned in the above module is burned in this ESP using Arduino and that is the code behind the functioning of the whole project.

## 4.3 Module 3

**MQTT (Message Queuing Telemetry Transport)** is an ISO standard (ISO/IEC PRF 20922) publish-subscribe-based messaging protocol. It works on top of the TCP/IP protocol. It is designed for connections with remote locations where a "small code footprint" is required or the network bandwidth is limited. The publish-subscribe messaging pattern requires a message broker as we are using mosquitto here.

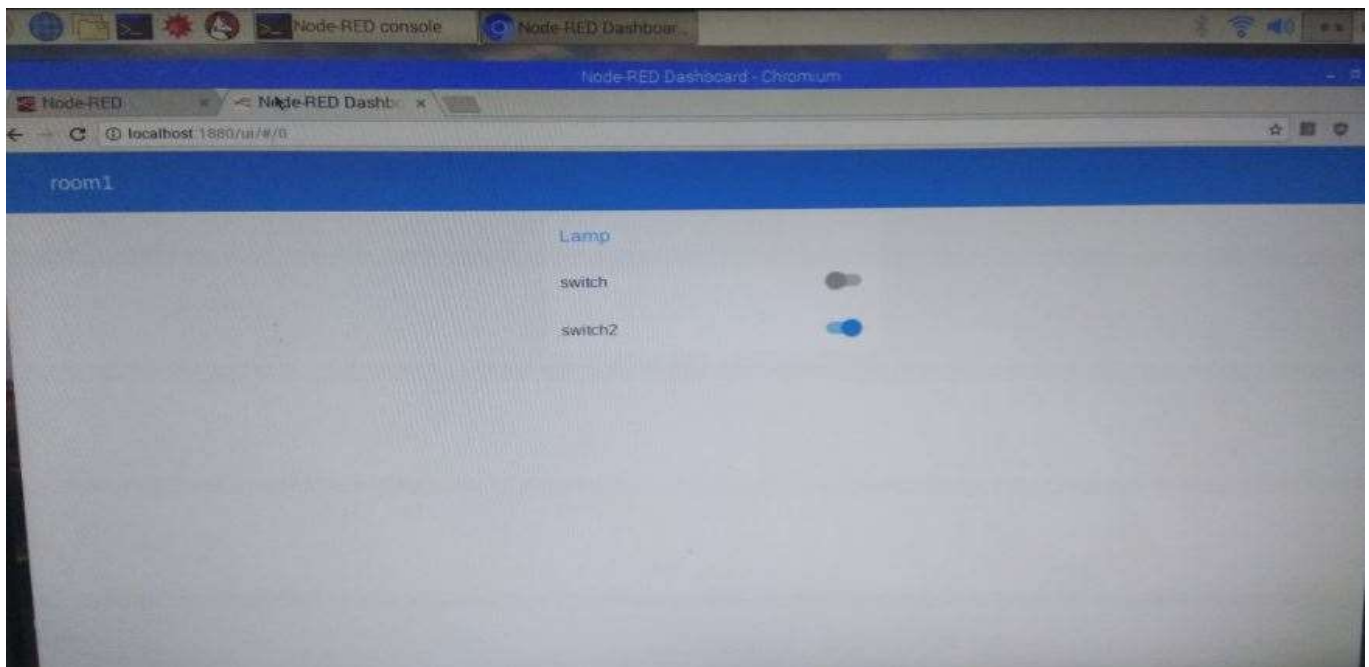
Two important terms and functions associated are:

- 4.3.1 Publish: Returns immediately to the application thread after passing the request to the MQTT client.
- 4.3.2 Qos: Quality of service refers to traffic prioritization and resource reservation control mechanisms rather than the achieved service quality. Quality of service is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow.

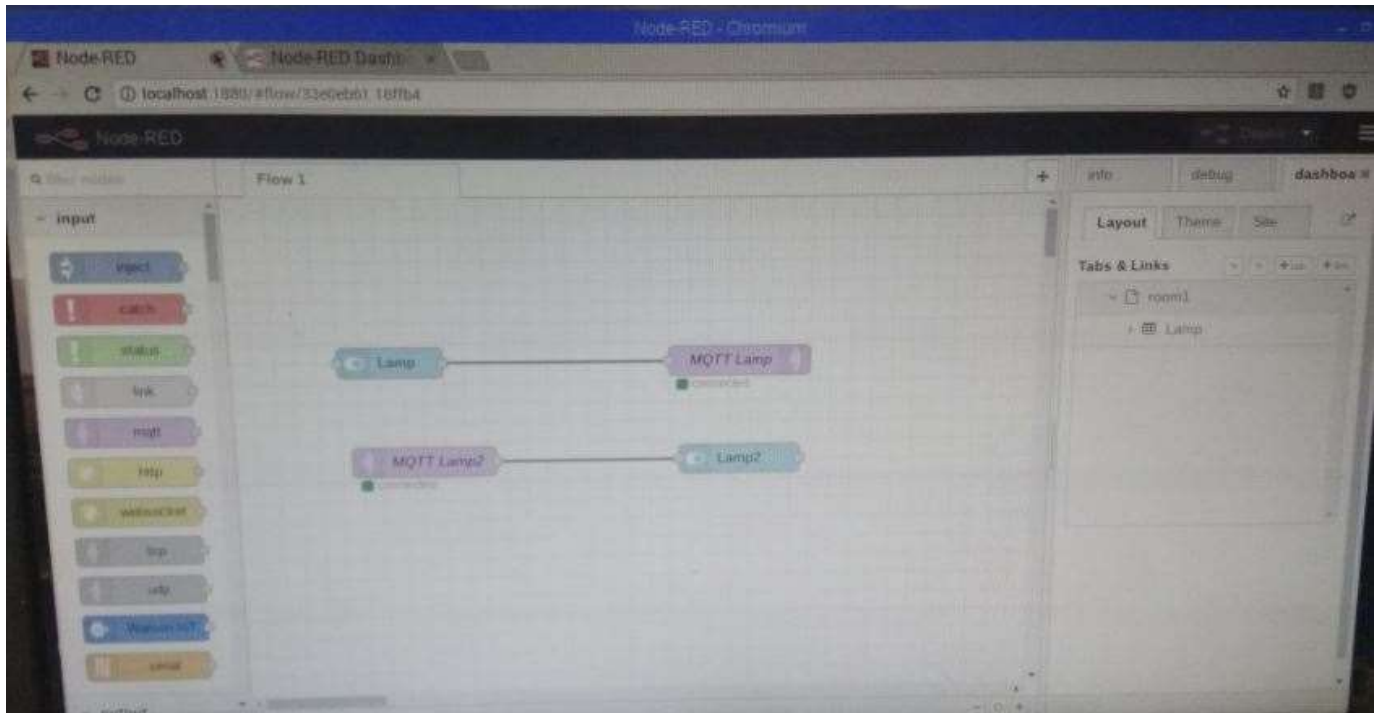
A description of each QoS level is found below:

- 1. At most once delivery (fire and forget)
- 2. At least once delivery (acknowledged delivery)
- 3. Exactly once delivery (assured delivery)

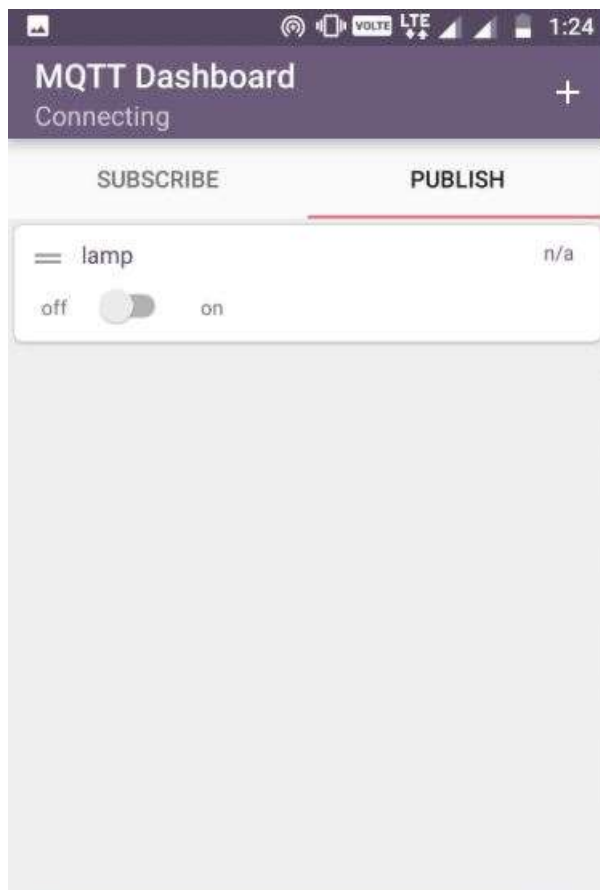
Here, we are using QoS value as 2.



**Figure 4.3.1: User Interface**



**Figure 4.3.2: MQTT Flow chart**



**Figure 4.3.3: MQTT Dashboard App**

#### 4.4 Module 4

The Hardware of the project looks something like this. The function of this switch is similar to the one used in MQTT dashboard and Node-RED. The switching “on” of the light in the switch shows that it is in on state and thus any appliance connected to it is also on and vice versa. Module 2 i.e ESP8266, a relay, an LED and a power supply is enclosed in this module physically.



**Figure 4.4: HARDWARE**

## **5.CONCLUSION**

At NIELIT, we took up training in the field of IoT for a period of 6 months where we were made familiar with the basic concepts and applications in the field, with focus on industry based applications. Along with this, the concepts regarding the fields of C Language, basic programming, software usage and its various applications were imbibed in our curriculum. KEIL software programming was also introduced using embedded C.

Tools like Proteus, 8051 Microprocessor programming, ARDUINO UNO, ESP8266 were used for making us familiar with the basic logics and their implementations in various practical fields and how is the hardware used. The actual usage started when we started making our project. We worked using Raspberry Pi, Node-Red, MQTT protocol and ESP8266 for the completion of our project. It helped us learn a lot many new things about all the hardware and software tools we used.

## **6.REFERENCES**

1. <https://projects.raspberrypi.org/en/projects/lamp-web-server/>
2. <https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT>
3. <http://www.circuitbasics.com/useful-raspberry-pi-commands/>
4. <https://www.raspberrypi.org/>
5. <https://en.wikipedia.org/wiki/ESP8266>
6. <https://en.wikipedia.org/wiki/MQTT>
7. Class lectures (Node-Red.pdf)