HOME AUTOMATION USING INTERNET OF THINGS

A PROJECT REPORT

Submitted by

MILAN MODI (212712106078)

KARTHIKEYAN K (212712106064)

NIZAM M (212712106092)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

SRI VENKATESWARA COLLEGE OF ENGINEERING

ANNA UNIVERSITY: CHENNAI – 600 025

APRIL 2016

ANNA UNIVERSITY: CHENNAI - 600025

BONAFIDE CERTIFICATE

Certified that this project report **"HOME AUTOMATION USING INTERNET OF THINGS"** is the bonafide work of **KARTHIKEYAN K, MILAN MODI,** and **NIZAM M** who carried out their project work under my supervision.

SIGNATURE

Dr. S. MUTHUKUMAR
HEAD OF THE DEPARTMENT
Department of Electronics and Communication Engineering,
Sri Venkateswara College of Engineering
Pennalur,
Sriperumbudur Taluk,
Tamil Nadu - 602 117

SIGNATURE

Ms. P. MADHUMITHA
SUPERVISOR
Assistant Professor
Department of Electronics and Communication Engineering,
Sri Venkateswara College of Engineering
Pennalur,
Sriperumbudur Taluk,
Tamil Nadu - 602 117

Submitted for the Project viva voice examination held on…….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

This project aims at a disparate approach to tackle the notion of home automation. The concept of Internet of Things has been embraced in order to tackle automation problem. The goal is to device a method to control the day to day household appliances in a way that is simplistic, comfortable, and easy to practice and understand by generic individuals. The availability of low cost microprocessors has been instrumental in developing this project. The salient feature of this project is granting the users complete dominion over their household appliances from anywhere in the world over the Internet. The state of the appliances can be monitored and changed according to the user's necessity.

ACKNOWLEDGEMENT

We express our sincere thanks to **Dr. S. Ganesh Vaidyanathan,** Ph.D., Principal, Sri Venkateswara College of Engineering, for providing us an opportunity to work on this project.

We express our sincere thanks to **Dr. S. Muthukumar,** Head of the department, Department of Electronics and Communication Engineering, Sri Venkateswara College of Engineering, for his constructive criticism during the reviews of this project.

We express our sincere thanks to our project coordinators **Ms. P. Jothilakshmi**, Assistant Professor, **Ms. S. Nandhini,** Assistant professor, **Ms. K. Thaiyalnayaki,** ,Assistant Professor, Department of Electronics and Communication Engineering, Sri Venkateswara College of Engineering, for their valuable suggestions towards the progress of this project.

We express our heartfelt thanks to my supervisor **Mr. P. Madhumitha,** Assistant professor, Department of Electronics and Communication Engineering, Sri Venkateswara College of Engineering, for his immense and timely help given during this project work. We also thank him for giving this opportunity to work under him. We would like to thank our entire department faculty members and friends for their valuable suggestions and support given during this project work.

TABLE OF CONTENT

PAGE NO.

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

IoT	**Internet of Things**
MQTT	**Message Queuing Telemetry Transport**
ARM	**Advanced RISC Machines**
CPU	**Central Processing Unit**
RAM	**Random Access Memory**
HDMI	**High-Definition Multimedia Interface**
DSI	**Digital Interface**
CSI	**Camera Interface**
USB	**Universal Serial Bus**
GPIO	**General Purpose Input Output**
SDIO	**Secure Digital Input Output**
SPI	**Serial Peripheral Interface**
SD	**Secure Digital**
OS	**Operating System**

UART	**Universal Asynchronous Receiver Transmitter**
QFN	**Quad Flat No-leads**
RF	**Radio Frequency**
RISC	**Reduced Instruction Set Computer**
PA	**Power Amplifier**
DCXO	**Digitally Controlled Oscillator**
MAC	**Media Access Control**
WEP	**Wired Equivalent privacy**
TKIP	**Temporary Key Integrity Protocol**
AES	**Advanced Encryption Standard**
WAPI	**WLAN Authentication and Privacy Infrastructure**
P2P	**Peer-to-peer**
SoftAP	**Software enabled Access point**
LNA	**Low Noise Amplifier**
MIMO	**Multiple Input Multiple Output**
STBC	**Space-Time Block Coding**
HTML	**Hyper Text Markup Language**
HTTP	**Hyper Text Transfer Protocol**
API	**Application Program Interface**

M2M	**Machine-to-Machine**
QoS	**Quality of Service**
IDE	**Integrated Development Environment**
CoAP	**Constrained Application Protocol**
REST	**Representational State Transfer**
AMQP	**Advanced Message Queuing Protocol**

1. INTRODUCTION

1.1 Internet of Things:

1.1.1 DEFINITION:

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.

The Internet of Things (IoT) is the network of physical objects—devices, vehicles, buildings and other items—embedded with electronics, software, sensors, and network connectivity that enables these objects to collect and exchange data.

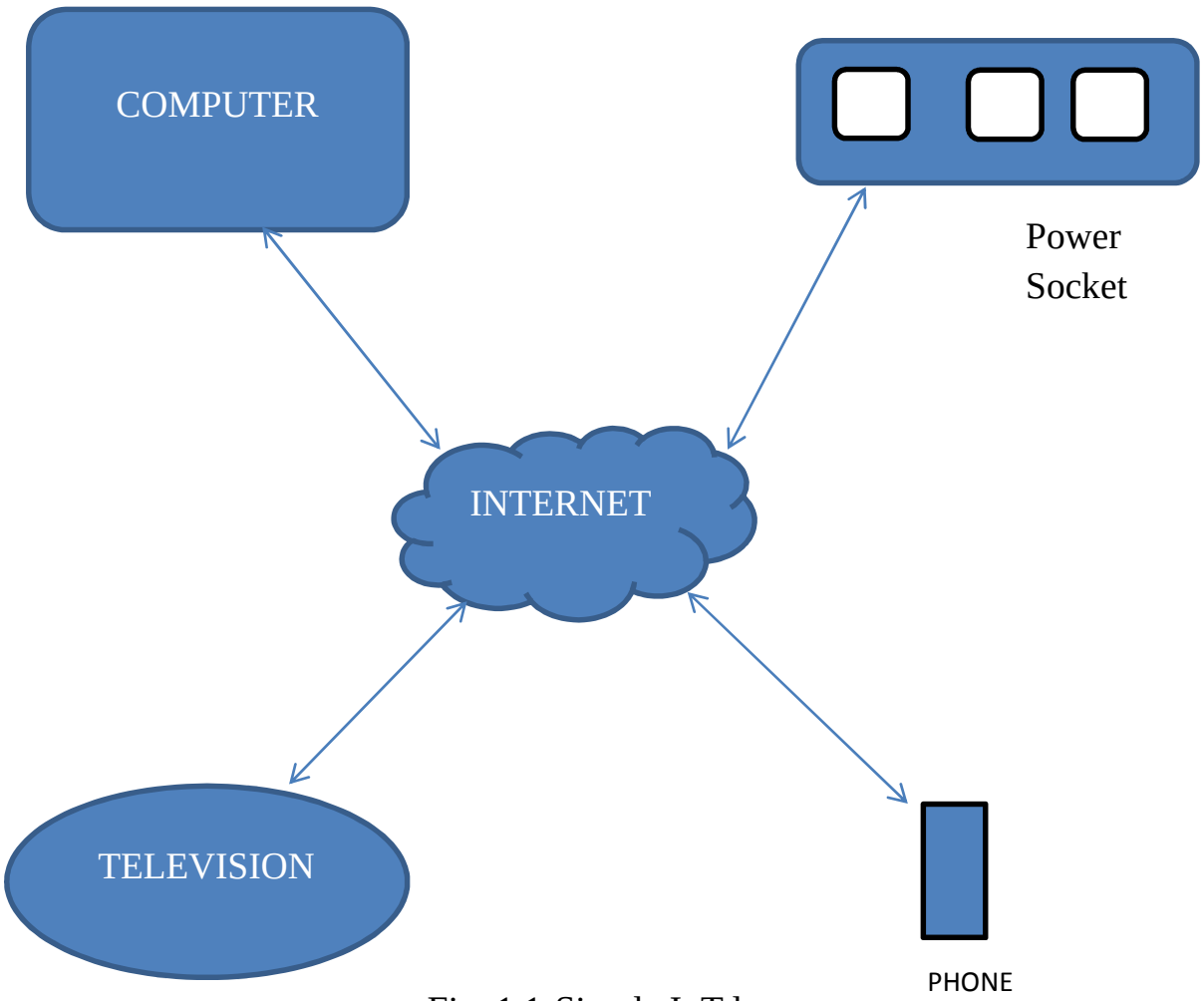

Fig. 1.1-Simple IoT layout

1.1.2 GENERAL:

This world is under the process of digitization. Interconnectivity amongst the electronic devices is preferred by any individual. A seamless, uninterrupted, movement of data between the devices and accessibility of the same is much expected, this is the ideal conundrum that can be tackled by Internet of Things. Conceptually, IoT is simple and easy to understand but nevertheless quite powerful. Imagine a scenario, where all objects in our mundane life were equipped with individual identifiers and wireless connectivity, these objects could very well be communicating with each other and can be supervised, and managed effectively by computers.

IoT depicts a system where the items of the physical world, comprising of either in-built sensors or external sensors attached to these items, are connected to the Internet via wireless and wired connections. It defines an architecture which comprises of specialized, dedicated hardware boards, Software systems, web APIs, and protocols all of which in conjunction creates a seamless environment. This customized environment allows embedded devices to connect to the internet and granting permission to access the sensory data and control systems that can be triggered over the internet. A “thing”, in the Internet of Things, may denote a person with a heart monitor implant, a sea creature with a chip transponder, an automobile with built-in sensors to alert the driver in the presence of low coolant in the radiators. In general a “thing” may represent any natural or man-made object that can be assigned with an IP address and provided with the ability to transfer data over a network.

1.2 RASPBERRY PI:

1.2.1 What is Raspberry Pi?

The domain of Electronics and circuit design has evolved by leaps and bounds in the recent decade and one such outcome of this burgeon is the design of this miniaturized computer called Raspberry Pi which is used as a developmental board. It is a low cost, credit-card sized computer that has the provision to plug into a computer monitor or TV, and works with a standard keyboard and mouse. Though small, this potent little device provides platform for people of diverse age groups to explore the art of computing, and to learn knack of programming in languages like C, C++, and Python. Its capability is equivalent to that of a desktop computer right from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

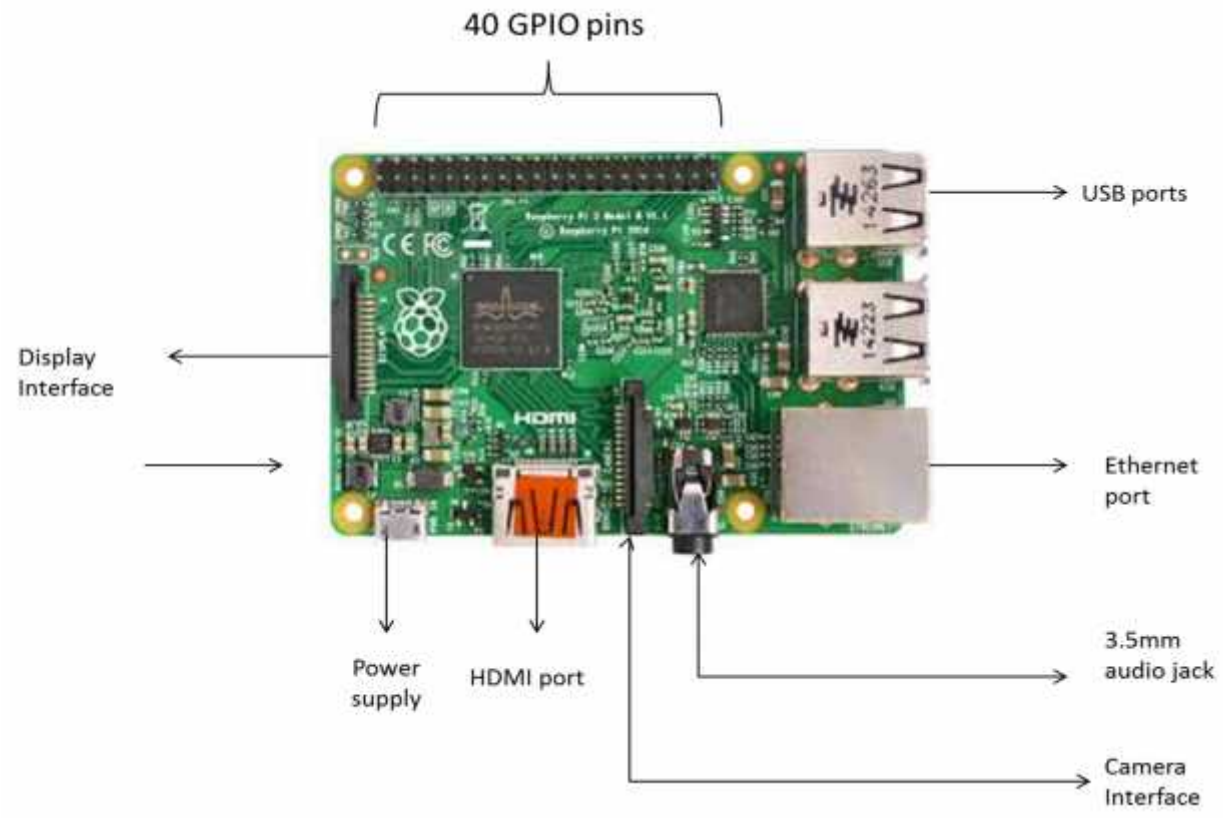

Fig. 1.2-RaspberryPi

1.2.2 How it operates?

The most striking feature is the Raspberry Pi's ability to interact with the outside world without having to conciliate on its processing capabilities and it is the reason why this developmental board has played instrumental roles in a wide array of recent projects.

Like any other computer Raspberry Pi also needs an OS to boot and run about with its normal functionality. There is a provision to mount a SD card into the Raspberry Pi and this acts as the equivalent to that of a hard drive seen inside any computer. Surprisingly, the size of the SD card need not be greater than eight Giga bytes for normal operation of the Raspberry Pi, depending on the user's storage needs a card of greater memory capability can be used, moreover it is inside this SD card that the Operating System necessary for the board's functioning is placed. There are several official distributions of OS for Raspberry Pi, also called as distros of which NOOBS installer is the easiest. Some of the other distros are Raspbian, Ubuntu, etc. most of which are Linux based Operating Systems. Amongst all the distros, Raspbian is highly recommended as it is designed specifically for Raspberry Pi.

1.2.3 Hardware support

There is a provision of connecting a HDMI cable so that any monitoring device with the capability to display via its HDMI port can serve as a monitor for Raspberry Pi; this again is an example of its versatility. Similarly USB port provisions are available for connecting other necessities such as keyboard, mouse, Bluetooth dongle, etc. inherently any USB device. The presence of an Ethernet port allows the Raspberry Pi to sustain a wired connection to access the Internet.

The number of USB ports available and other hardware features depend on the model and the latest version presents an on-board Wi-Fi and Bluetooth hardware.

1.2.4 SPECIFICATION:

- A 900MHz quad-core ARM Cortex-A7 CPU
- 1GB RAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- 3.5mm audio jack
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot
- VideoCore IV 3D graphics core

1.3 NODEMCU:

1.3.1 Development Board

NodeMCU is an Open Source Firmware Development Board that helps to mainly build IoT projects with the help of few LUA Scripts. Usually the development board consists of a pre-loaded LUA scripting language, but it is also compatible with Arduino IDE and can be programmed in a fashion similar to that of Arduino. It is analogous to Arduino Hardware with a built in Input / Output pins, it possesses a chip with built in Wi-Fi that enables direct connectivity to internet, thereby facilitating easy control of things from online. It accelerates the Internet of Things projects by providing simplistic and easy development environment for the programmers. The Development Board is based on ESP8266 Chip, integrated GPIO (General Purpose Input Output), PWM (Pulse with Modulation), IIC (Interconnected Integrated Circuit), and ADC (Analog to Digital Converter). It supports a wide variety of libraries, including those for HTTP, UDP, MQTT, and normal GPIO controls.

1.3.2 ESP8266

The ESP8266 chip is the primary component of the development board, it works on 3V logic. It is the chip that is chiefly responsible for the board's ability to connect to Wi-Fi. The early models possess a self-contained System on Chip, and a full TCP/IP stack thereby providing any microcontroller connected to it, the ability to access the Wi-Fi networks. Owing to the technological developments, the recent models have microcontroller ability integrated with the Wi-Fi chip. It has a powerful on-board processing unit and the storage capability is sound enough to allow integration with sensors and other applications via the GPIO pins.

1.3.2.1 Features

- SDIO 2.0, SPI, UART
- 32-pin QFN package
- Integrated RF switch, 24dBm PA, DCXO, and PMU
- Integrated RISC processor, on-chip memory and external memory interfaces
- Integrated MAC/baseband processors
- Quality of Service management
- I2S interface for high fidelity audio applications
- Integrated WEP, TKIP, AES, and WAPI

1.3.2.2 Specifications

- 802.11 b/g/n
- Wi-Fi Direct (P2P), SoftAP
- Integrated TCP/IP protocol stack
- Integrated T/R switch, LNA, power amplifier and matching network
- Integrated PLLs, regulators, DCXO and power management units
- +19.5dBm output power in 802.11b mode
- Power down leakage current of <10uA
- Integrated low power 32-bit CPU could be used as application processor
- SDIO 1.1/2.0, SPI, UART
- STBC, 1×1 MIMO, 2×1 MIMO
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW

1.4 MQTT:

The reduction in the size of the device has placed immense constraints on the power consumption and memory availability. The yester year protocols were too heavy and complex, providing minimal optimization for the presented hardware. Therefore a new protocol with light and simple architecture was long overdue and so MQTT was framed. MQTT (Message Queuing Telemetry Transport) is a message oriented machine-to-machine (M2M) connectivity protocol.

Designed as an extremely lightweight publish/subscribe messaging transport, it is useful for connections with remote locations. The striking feature is the protocol's ability to be employed even along with hardware which can comply only with small code footprint. It is also employed in areas when the bandwidth is limited and premium. For instance, it has been used in sensors communicating to a broker via satellite link and small device scenarios where there is a constraint on the bandwidth availability and necessity to reduce the excess processing capability in the hardware. The latest version is MQTT Version 3.1

It is also ideal for mobile applications because of its small size, low power usage, minimized data packets, and efficient distribution of information to one or many receivers. Yet another point worth highlighting is the fact that the protocol is very energy efficient and draws minimal power for operation. Though invented as early as 1999, the full potential of the protocol has been unleashed owing to the drastic improvement and technological advancement in the field of electronics. Currently the protocol is undergoing the process of Standardization and the port 1883 has been designated as the standard port.

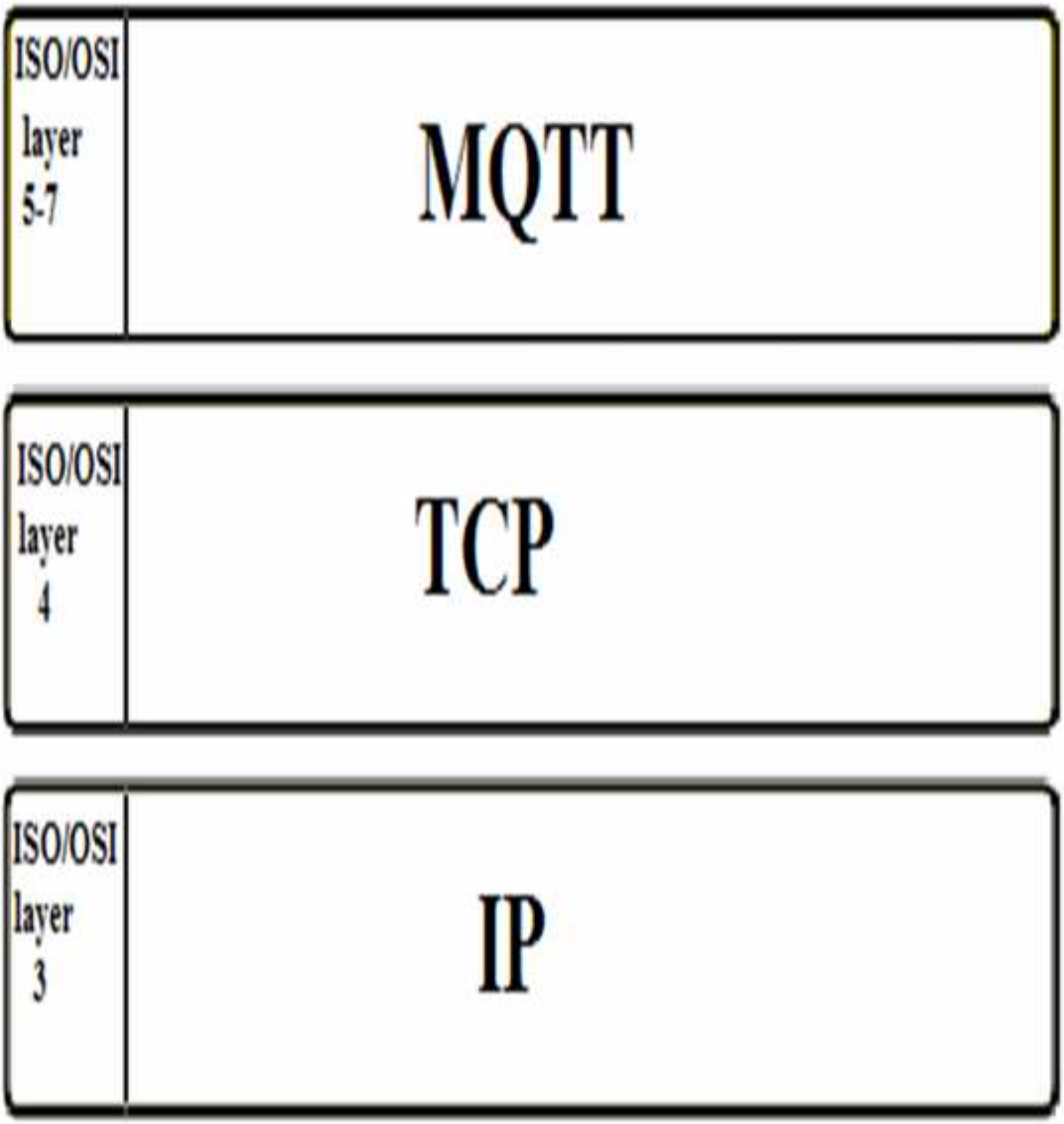

Fig. 1.3-MQTT layers

1.5 Web Server

1.5.1 General

Any web page that is accessible or viewable over the internet is inherently hosted by a Web Server. In other words, without Web Servers there will not be any Web pages to browse over the Internet. The images, videos, GIF or any text document or PDF that is available on a remote computer must be served and made accessible to the users via Web pages hosted by the Web Servers. Web servers mainly respond to requests initiated by the user anywhere in the world accessed via the World Wide Web. They generally process the HTTP requests that are made by the client via the various and numerous network protocols. It is responsible for serving the web pages to the users on their demand, storing the web pages and also processing the same. The entire communication between the clients and the server is based on the Hyper Text Transfer Protocol. Most times the pages that are delivered or served are HTML documents, which can comprise data ranging from text, images, scripts, etc.

The clients communicate with the web servers using web browsers in most cases. The request for a particular page is initiated by the web browser and the server delivers the same. Though the primary purpose of the web server is to serve web pages and user requested data, it is also capable of receiving data from the user, for instance, uploading a file. This is owing to the ability of the HTTP protocol. Web servers also support scripting languages such as PHP, java script, etc. This facilitates the user to control the server’s behavior using a set of scripts. Servers need not necessarily be accessed via the World Wide Web; they can also be deployed to be accessed only through the local networks which can be used for printers and other devices. Multiple servers can be employed in case of heavy traffic requests.

1.5.2 Flask

It is quite difficult for someone to create a web server from the very scratch moreover it is time consuming, and so many vendors set up a basic frame layout that serves as a guideline in order to ease the process of creating a web server. Many such vendors facilitate creating and hosting of a custom web server catering to our specific needs. Some of the vendors are apache, njinx, etc. Flask, like the other vendors is used to create our custom web servers but with a slight variation from the others. Flask is mainly a web framework that can be written using the python language. Web framework is used for development of web applications such as web servers, web APIs, etc.

Flask is mainly server-side web framework, enabling us to display our web pages and applications over the server. Flask doesn't force the users to use a particular set of library or tools, thereby enabling easy and fast development of web applications. Like any other web server development tool, Flask can serve up static files, scripts, regular HTML documents, images, etc. It has the functionality of creating a developmental web servers for dry run purpose and can has an option of running the server in debug mode, if the user wants to make a log of the crash details. Major advantage of Flask is the fact that it is compatible with many extensions that caters specific needs and purpose and it can process RESTful request. As it is a micro-framework there are little or no dependencies on external libraries.

1.6 WebScokets

When visiting a typical web page, each resource like image, css, javascript file, etc. that the browser comes across when interpreting the document creates a new request to the server using the HTTP protocol. This includes requests made using Javascript and HTTP Requests.

1.6.1 What is WebSocket?

WebSocket is a protocol providing full-duplex communication channels over a single TCP connection. WebSocket is designed to be implemented in web browsers and web servers, but it can be used by any client or server application. Being an independent TCP-based protocol its only relationship to HTTP is that its handshake is interpreted by HTTP servers as a request that can be upgraded.

1.6.2 Conventional protocol

"Protocol" is an established set of conventions. In the case of a HTTP protocol, it is a convention for connecting, transmitting, receiving, and disconnecting. WebSockets use a different protocol from HTTP, one that makes a connection and then keeps it open until you leave the page. Though WebSockets use a different protocol, they initiate communication over HTTP and then transparently change to the special WebSockets protocol. Communication over this connection is not the same as the HTTP GET/POST/etc. it's a special protocol that supports simultaneous data transfer in both directions.

1.6.3 Difference between HTTP and WebSocket

Consider a server and client has established connectivity over traditional HTTP by handshake process. Each time when the client requires some data from the server, it has to initiate a full-fledged HTTP request. This process is time consuming and if

the request takes long time to process the user will have to reset the connection by initiating the handshake again. Sometimes the server may deliberately send the data slowly to avoid such handshake process.

But WebSockets are more like a two-way communication between the server and client. Once a handshake has been established for transferring the data it ensures that the channel/connection is kept open in case the client needs any other data form the server. The connection is kept open until any one of the two initiates a release. This enables simultaneous communication and transfer of any format of data even user defined data.

Fig. 1.4-WebSocket communication

The server pushes the data to the web browser; the transmission and reception of data occur simultaneously. Since the channel is kept open throughout the entire conversation, the exchange of data doesn’t suffer any connection loss as in HTTP protocol.

2. Existing System

The system that is most widely prevalent around the world is empowering the user to control the appliances in and around him by using IR remote. Yet another variant is controlling the appliance by sending a short message from your phone by availing the SMS service provided by GSM. In some places, for controlling the appliances over a short range Bluetooth technology has been employed thereby providing access to the appliances via any smart phone equipped with such facility. Though these methods have their own distinct advantages, they are all unified by their common setback, namely their inability to access appliances over long distance. Each of the methods stated above are constrained by the range of the components and devices used in crafting the system. The range of an Infra-Red remote is limited to a maximum of few meters and the connectivity range of Bluetooth is at most around 10 to 20 meters. This limitation makes dynamic control and accessing of appliances difficult and next to insurmountable. Though GSM method of accessing may provide long range communication, the amount of data spend in transmitting and receiving information via SMS and the repeated to and from communication necessary to receive the status of the appliance makes this method a little cumbersome and time consuming not to mention the latency involved in reception of SMS in case of poor cellular signal.

3. Proposed System:

3.1 LITERATURE SURVEY:

Mrs. Shubhangi A. Shinde, in her paper on "**MQTT-Message Queuing Telemetry Transport protocol**"published in International Journal of Research gave a fundamental overview about the concept of MQTT protocol. As mentioned earlier, MQTT is a Client Server publish/subscribe messaging transport protocol. It is light weight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium. The protocol runs over TCP/IP, or over other network protocols that provide ordered, lossless, bi-directional connections. As per MQTT V3.1 Protocol Specification, "MQ Telemetry Transport (MQTT) is a lightweight broker-based publish/subscribe messaging protocol designed to be open, simple, lightweight and easy to implement." MQTT runs over TCP/IP. It enables transfer of telemetry style data which is nothing but sensor and actuator data. The sensors and actuators communicate with applications through MQTT message broker. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium. For example: Usage in health clinics where doctors can remotely monitor patients at their home. The key features mentioned by Mrs. Shubhangi are

- Use of the publish/subscribe message pattern which provides one-to-many message distribution and decoupling of applications.
- A messaging transport that is agnostic to the content of the payload.

Mr. Konglong Tang professor in the Ocean University of China published a paper in **Design and Implementation of Push Notification System Based on the MQTT Protocol** proposing that the use of MQTT with support of a server message server can put forward solutions for the server sending the specified messages to the specified mobile devices and the mobile clients receiving pushed messages from the server. The purpose of this technology is to make full use of the Internet, push the different areas of information to the company's customers, such as science technology, economics, sports, education. He further adds that, in after a few years, the push technology has become the most popular network of technology, many companies are aiming at the technology for research, and launched their own products, and apply this technology to many operating systems. The overall design is based on subscribe/publish feature of MQTT. The design is based on publishing/subscribing messaging mode .In this model, the publisher and subscriber are both clients, the message destination is called theme. By connecting to the message broker they transfer data across the network. Publishers send a specific topic of messages to message broker, subscribers subscribe specific news topics to the message broker, and the connection between the subscriber and publishers managed by the message broker. When the message broker receives the published messages, it delivers the message to subscriber. Publishing/subscribing messaging model allows multiple providers to publish messages to the same topic. It also allows multiple users to subscribe messages with a subject. Then the message broker will broadcast to different subscribers. This concept enhances the efficiency of the server running MQTT.

3.2 Overview

Though the system in existence is used in many places, they are limited in functionality and the range of connectivity by form of communication employed. The aim of this project is to make an automation system specifically catering to control the house hold appliances, without comprising the currently existing system's drawbacks. The concept of Internet of Things is exercised in order to create an automation system that is not bound or limited by the physical range of the device or the communication type employed.

The medium for establishing communication between the user and the appliances at home is The Internet. All the devices present in the house hold are controlled through the Internet by accessing a specific webpage which displays the appliances present in the house. The system is flexible enough to allow the user to either turn ON or OFF the appliance. The Internet like the physical world; is a vast region, having said that it is mandatory for the users to be able to identify themselves and others in this vast region if at all they were to communicate amongst themselves. It becomes mandatory for us to follow certain addressing format if we are to identify the position of the devices with utmost swiftness and accuracy. Internet Protocol does this job by providing address to devices connected to the Internet. Any device that is connected and accessible via the Internet must inexplicitly have its own IP address. It is this IP address that acts as the indicator as to where the device is located analogous to how the address of our home defines our presence in a particular location in space and time.

Therefore by providing IP address to every device that needs to be automated inside the house, user is provided the ability to control the devices via Internet. IP addresses are quite difficult to be assigned to a device that does not possess a

provision to process data, such as a microcontroller or microprocessor. For instance, a fan is a device with a primary functionality of either being switched ON or OFF; therefore there is no need for a microcontroller to be placed within it. But in order to connect it over to Internet it is mandatory that we assign an IP address to the fan. The solution is pretty straight forward; connect the fan via a relay to a microcontroller that is capable of accepting an IP address and can receive/ transmit data over Internet.

Any user who wants to access the Internet will have to use a Web browser. In order for a web page to be available for the user, it must be served or hosted by another party. Basically, any web page that users visit on the Internet is being hosted by some other user. The same principle applies for our project, if we are to provide ability to control the home appliances over the Internet; it is mandatory that we create a web page and host the same by using a Web server. Most of the Web Server that is seen over the Internet works on the request-response HTTP protocol. The web page is hosted via a HTTP servers namely because they provide better protection and security from external phishing and brute force attack. There are many ways to host a Web service, for instance, use web server vendors like apache, Flask, Django etc. Considering the fact that every house hold is vaguely similar, it is obvious that the web page must be crafted catering to each user's needs. Conventionally HTML language is preferred to creating the web page coupled with JavaScript in order to facilitate exchange of data between the server and the client namely the web browser.

3.3 BLOCK DIAGRAM:

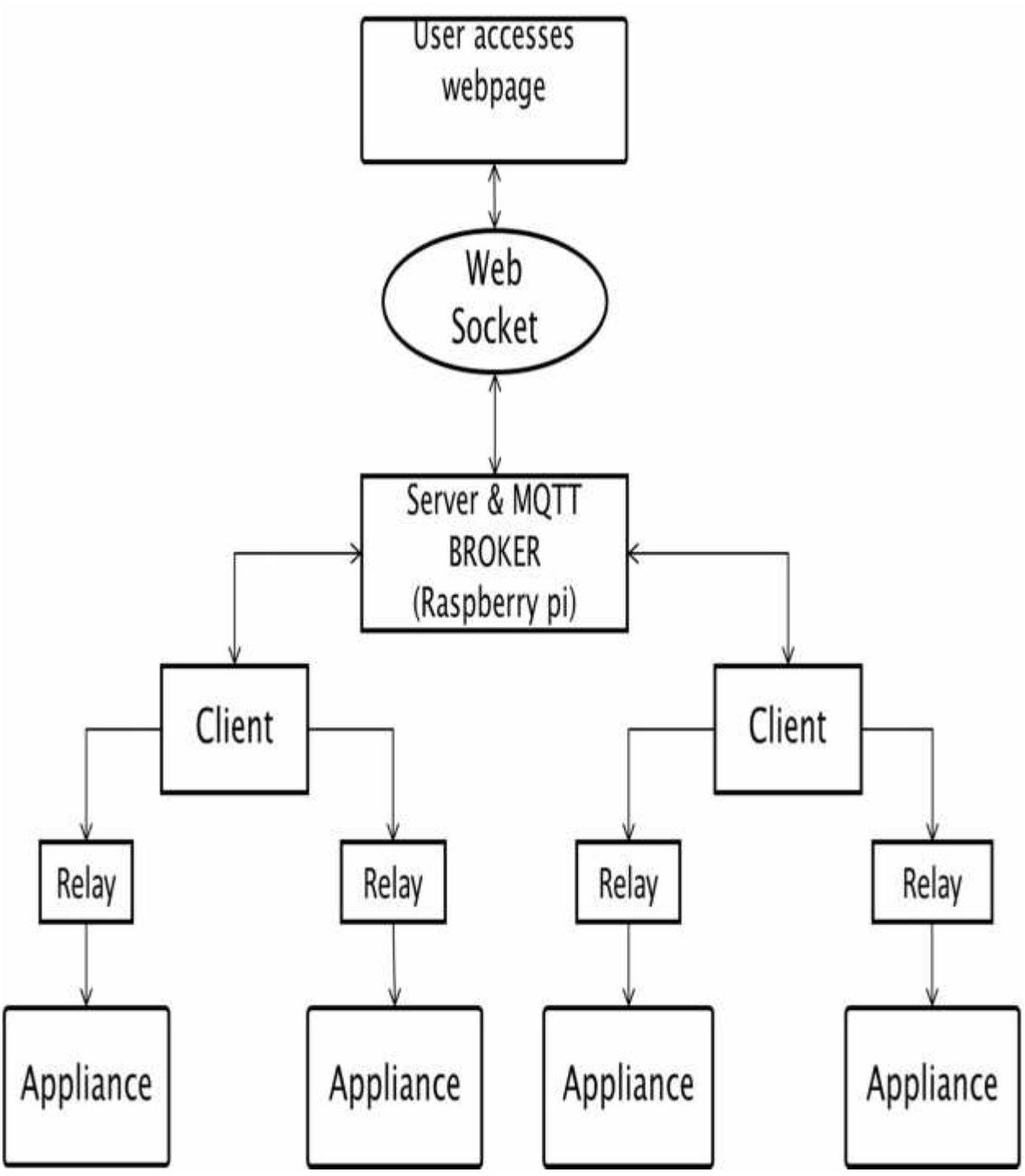

Fig. 3.1-Block diagram

3.4 Working

The user accesses the Web page that is served by a typical HTTP Web Server. Depending on the need the user might toggle an appliance ON or OFF. This data is transmitted to the MQTT broker. Both the HTTP server and MQTT broker are hosted from Raspberry Pi which is in turn connected to the Internet either through wired Ethernet cable or over Wi-Fi. Since there is not a direct channel or methodology to transfer data from a Web page working in HTTP protocol, to a broker working on a different protocol namely MQTT, we transfer the data by means of a WebSocket. The clients here refer to programmable microcontroller devices with capability to connect to Wi-Fi. They are connected to the home Wi-Fi network and IP addresses are provided to them on successful connection to Wi-Fi. The clients subscribed to particular topics receive messages from the same; on receiving a specific set of messages the client is programmed to trigger certain GPIO pins high. By connecting relays to the above stated devices we can control the triggering of any appliance. In short, we are providing IP addresses to the appliances indirectly by channeling it through the client devices. The clients are programmed to work on MQTT protocol and can be programmed to inter-communicate amongst themselves. In order for the clients to receive message from the broker, they need to unsubscribe from the specific topics namely the appliances. The clients are programmed with QoS 0.

3.5 Hardware

3.5.1 Raspberry Pi:

Raspberry Pi provides all the core functionality of a desktop computer at meager power consumption. The HTTP web server and the MQTT broker both are hosted using the Raspberry Pi booted with an OS that Linux based namely Raspbian. The device must be connected with the Internet in order for the server and broker to go online. Any Linux based system can be programmed to act and serve a Web page and host a broker by using proper libraries. The main advantage of choosing Raspberry Pi is that its efficiency, easy to handle and consumes low power. Unlike other computers which will require full-fledged external input devices in order to visualize what operation is being carried out, the Raspberry Pi can be accessed from any other computer over the same network by exploiting the Secure Socket Shell protocol. A Graphical User Interface of the Raspberry pi's desktop can be visualized on this host computer thereby providing easy debugging capabilities for the user. Being the host of such profound functionalities Raspberry pi has been chosen to perform the core operation of this project.

3.5.2 NODEMCU

The clients shown in the block diagram refer to the devices called as NodeMCU. It is a developmental board integrated with ESP8266 chip. Owing to the facilities of the chip, the developmental board has the ability to connect to a Wi-Fi network and is loaded with microcontroller capabilities. They can be programmed to perform a specific dedicate purpose and the program is stored in a flash memory seen on-board the chip. The board is in turn connected to the relay through one of its GPIO pins; controlling the relay's triggering mechanism.

3.5.2.1 Advantage

The main advantage of using NodeMCU is the presence of On-board Wi-Fi module thereby avoiding unnecessary additional components or Ethernet cables in order to obtain an Internet connection. Moreover the board is specifically designed to cater such needs; this is quite evident from the fact that chip size is minimal and optimal, the power consumption is very meager, the board can be initiated into a deep sleep mode to save power and most importantly it contains a TCP/IP stack enabling the board to support any protocol functioning with TCP/IP as its base.

Fig. 3.2-NodeMCU

3.5.3 Relay:

3.5.3.1 Working

A relay is analogous to an electronic lever that operates both electrically and mechanically; it triggers with a small current and in effect it triggers another, much larger current carrying circuit. They are used widely owing to the fact that many sensors produce only tiny electric current and it is impossible to bridge the gap in current to drive bigger machinery. Electromagnet is at the main component of a relay and it performs the switching mechanism.

There are two basic type of relay switching namely

- Normally Open Contact (NO) – Also called make contact. It closes the circuit when the relay is activated. It disconnects the circuit when the relay is inactive.
- Normally Closed Contact (NC) –Also called as break contact. When the relay is activated, the circuit disconnects. When the relay is deactivated, the circuit connects.

The working of a five pin relay is quite simple; when current flows through the control input terminal, the inductor coil gets energized and the electromagnet attached to the movable slider is attracted towards the coil and closes the switching circuit.

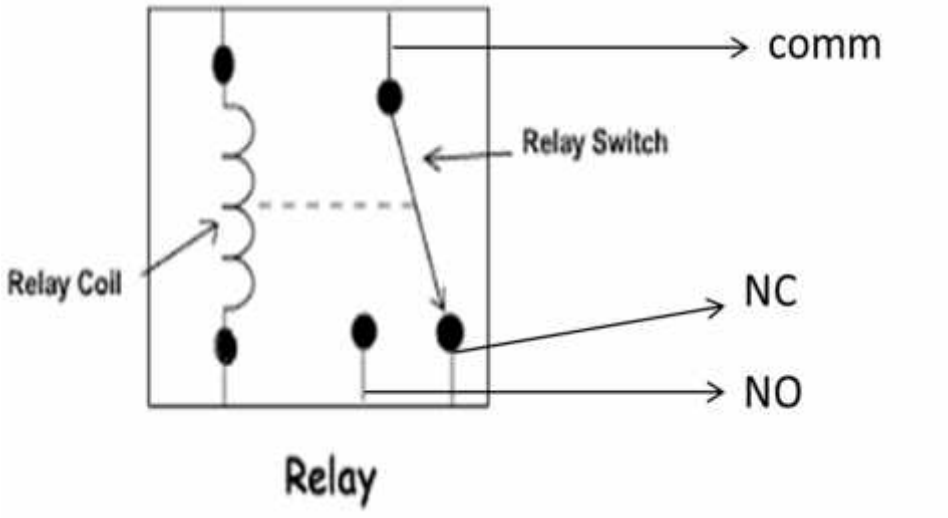

Fig.3.3-Basic relay structure

3.5.3.2 Advantage

The control input for the relay from the microcontroller is fed as input into to the darlington pair IC (ULN 2003); the output of which is given as relay's input. This is done because ESP8266 works on 3.3V logic and might not muster enough current to drive a 5V relay, so we use the darlington pair to provide sufficient current to drive the relay's coil.

Fig. 3.4-Relay module with ULN2003 IC

3.6 Protocol

3.6.1 MQTT

3.6.1.1 Clients and Servers

MQTT has a client/server model, where every sensor is a client and connects to a server, known as a broker, over TCP. Being a message oriented protocol, every message is transmitted as discrete chunks of data, and also ensuring it is opaque to the broker. Messages are published to an address, also known as a topic. Clients are able to receive the messages by subscribing to these addresses or topics, furthermore client may subscribe to more than a single topic. Every client subscribed to a topic receives every message published to that specific topic. Unlike other protocols the topics in MQTT are arranged in a hierarchical manner resembling that of a filing system. Like many Linux and UNIX based systems, Wildcards are allowed in MQTT especially when registering a subscription allowing the clients to view the entire hierarchies of the topics present. MQTT supports three quality of service (Qos) levels, “Fire and forget”, “delivered at least once” and “delivered exactly once”. The clients can decide a custom “last will and testament” message to be sent by the broker in the event of disconnection of the client. These messages signal to the subscribers regarding details of the device disconnections. MQTT has the functionality and support for storing the persistent messages onto the broker. For example, when publishing messages, the client may request the broker to persist the message, but only the most recent message can be persisted and stored by the broker. The chief advantage of this functionality is whenever a client subscribes to a topic, any persisted message will be sent to the client. MQTT brokers can be programmed to ask for username and password from any client that is about to connect to it in order to increase the security and prevent

unauthorized access. The clients must be authenticated by the broker if the client wants to establish connection with the broker. If there is a necessity to increase the security and ensure privacy, the TCP connection may be encrypted with SSL/TLS.

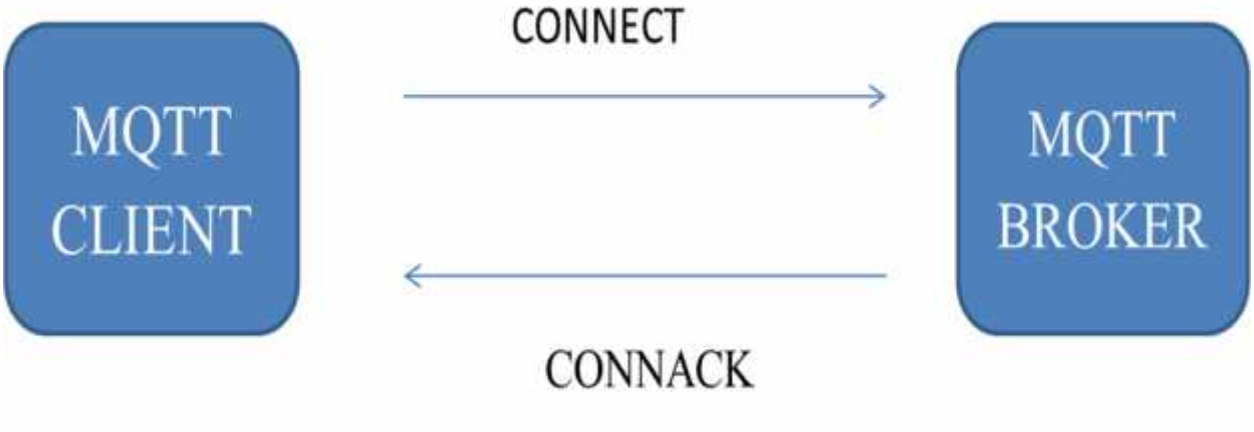

Fig. 3.5- MQTT Client and broker link

The clients can act as a publisher and also a subscriber simultaneously. The MQTT client in general can be any device from a microcontroller up to a full-fledged server containing an active running MQTT library and is connecting to an MQTT broker over any kind of network. It could be a small and resource constrained device that is connected over a wireless network and has a minimum functionality library strapped to it or a typical computer running a graphical MQTT client for testing purposes. Basically any device that has a TCP/IP stack and speaks MQTT over it. The client side implementation of the MQTT protocol is very straight-forward and really reduced to the bare essentials.

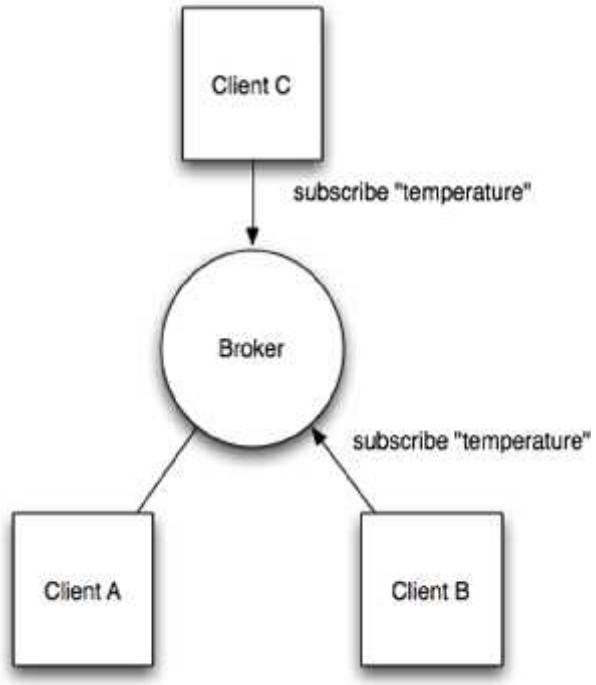

Fig. 3.6-Simple broker structure

Consider three Clients namely ‘A’, ‘B’, ‘C’. They are connected to a common broker which supports MQTT protocol. The clients which subscribed to a particular topic say “temperature” receive messages published to that topic. The topic serves the functionality of the address thereby facilitating the broker on to which clients the data must be delivered.

3.6.1.2 MQTT broker

The MQTT broker forms a bridge between the various clients that are using the MQTT protocol. Every client that is publishing the data must be connected to the broker in order to establish the channel for communication. The data is transmitted amongst the clients because of the broker is able to analyze and direct the data to the respective and intended user by having track of which topic is subscribed by the individual clients. These topics basically act as the address of the clients and provide detailed information for the broker on which of the clients are the intended recipient for the transmitted message. There are several brokers available for the implementation of the MQTT protocol such as mosquito, HiveMQ, rabbitMQ, ActiveMQ, etc. Some of the methods used in MQTT are

Control packet	Direction of flow	Description
CONNECT	Client to Server	Request to Connect to server
CONNACK	Server to client	Connection Acknowledged
PUBLISH	Client to Server and vice versa	Publish message
PUBACK	Client to Server and vice versa	Publish Acknowledged
PUBREL	Client to Server and vice versa	Publish Released
PUBREC	Client to Server and vice versa	Publish received

PUBCOMP	Client to Server and vice versa	Publish complete
SUBSCRIBE	Client to Server	Subscribe request
SUBACK	Server to Client	Subscribe Acknowledge
UNSUBSCRIBE	Client to Server	Unsubscribe request
UNSUBACK	Server to Client	Unsubscribe acknowledgement
PINGREQ	Client to Server	PING request
PINGRESP	Server to Client	PING response
DISCONNECT	Client to Server	Client is disconnecting

Tabulation 3.1-MQTT methods

3.6.1.3 QUALITY OF SERVICE

Quality of service determines how each MQTT message is delivered and must be specified for every message sent through MQTT. It is essential to choose an optimal value for QoS as it is vital in determining how the server and the client communicate and transfer message amongst themselves. Three QoS for message delivery could be achieved using MQTT:

- QoS 0 (At most once) - where messages are delivered according to the best efforts of the operating environment. Message loss can occur.
- QoS 1 (At least once) - where messages are assured to arrive but duplicates can occur.
- QoS 2 (Exactly once) - where message are assured to arrive exactly once.

There is a simple rule when considering performance impact of QoS. It is **“The higher the QoS, the lower the performance"**. MQTT provides flexibility to the IoT devices, to choose appropriate QoS they would need for their functional and environment requirements.

3.6.1.4 MQTT ARCHITECTURE:

There are two dominant data exchange protocol architectures:

1**. Broker based**

In this architecture, Broker controls the distribution of information. It stores, forwards, filters and prioritizes public requests from the publisher client to the subscriber clients. Clients switch between publisher role and subscriber roles depending on the functionalities desired.

Examples:

- AMQP
- CoAP
- MQTT

2. Bus based

In this architecture, clients publish messages for a specific topic which are directly delivered to the subscribers of that topic. There will not be any centralized broker or any broker based services here. Examples:

- DDS
- REST
- XMPP

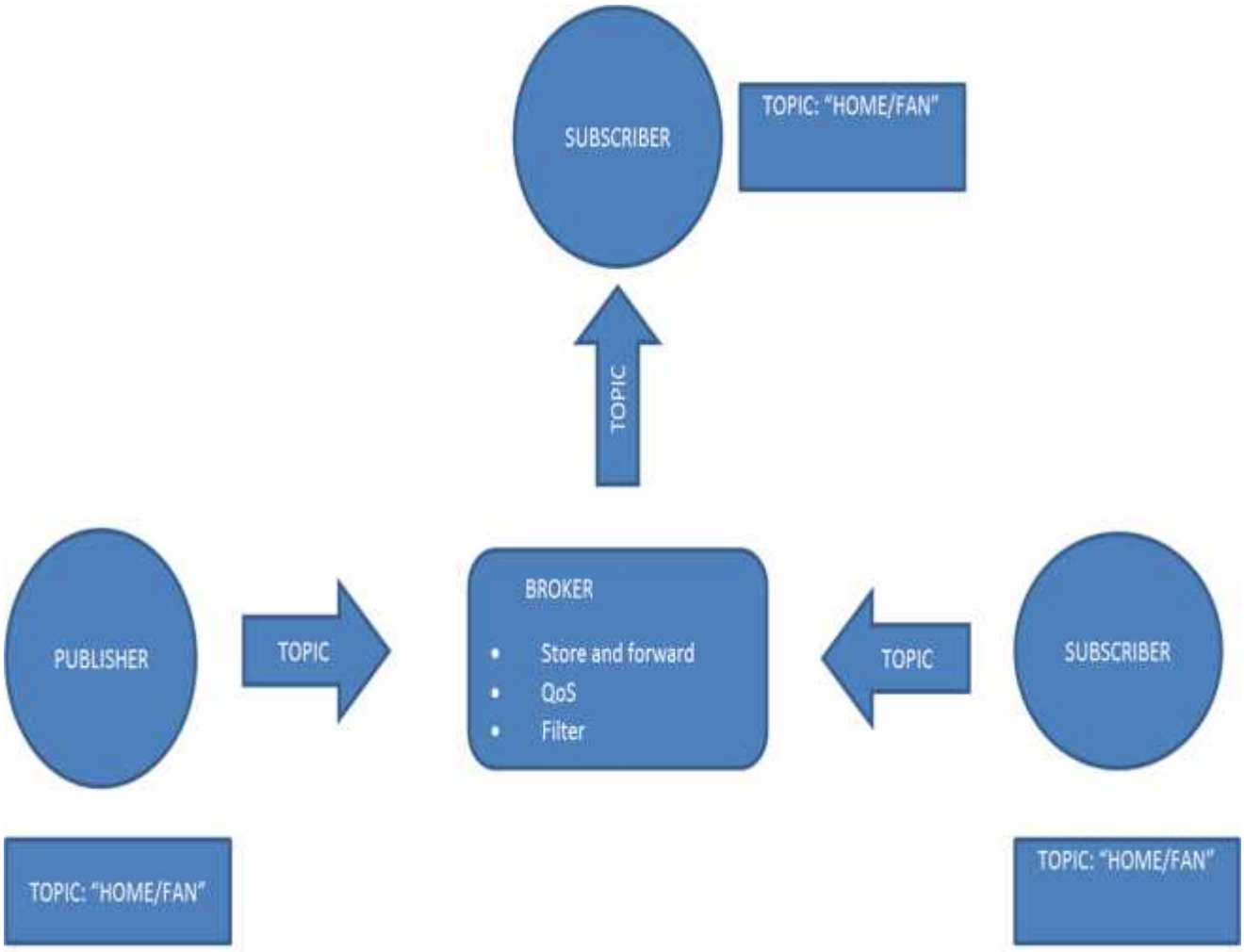

Fig. 3.7-MQTT Architecture

3.6.1.5 MQTT MESSAGE FORMAT:

MQTT messages contain a mandatory fixed-length header of 2 bytes and an optional variable length header specific for messages and message payload. In general, Optional fields complicate protocol processing. However, MQTT is optimized for bandwidth constrained and unreliable networks; typically wireless networks; so optional fields are used to reduce data transmissions as much as possible. MQTT uses network byte and bit ordering.

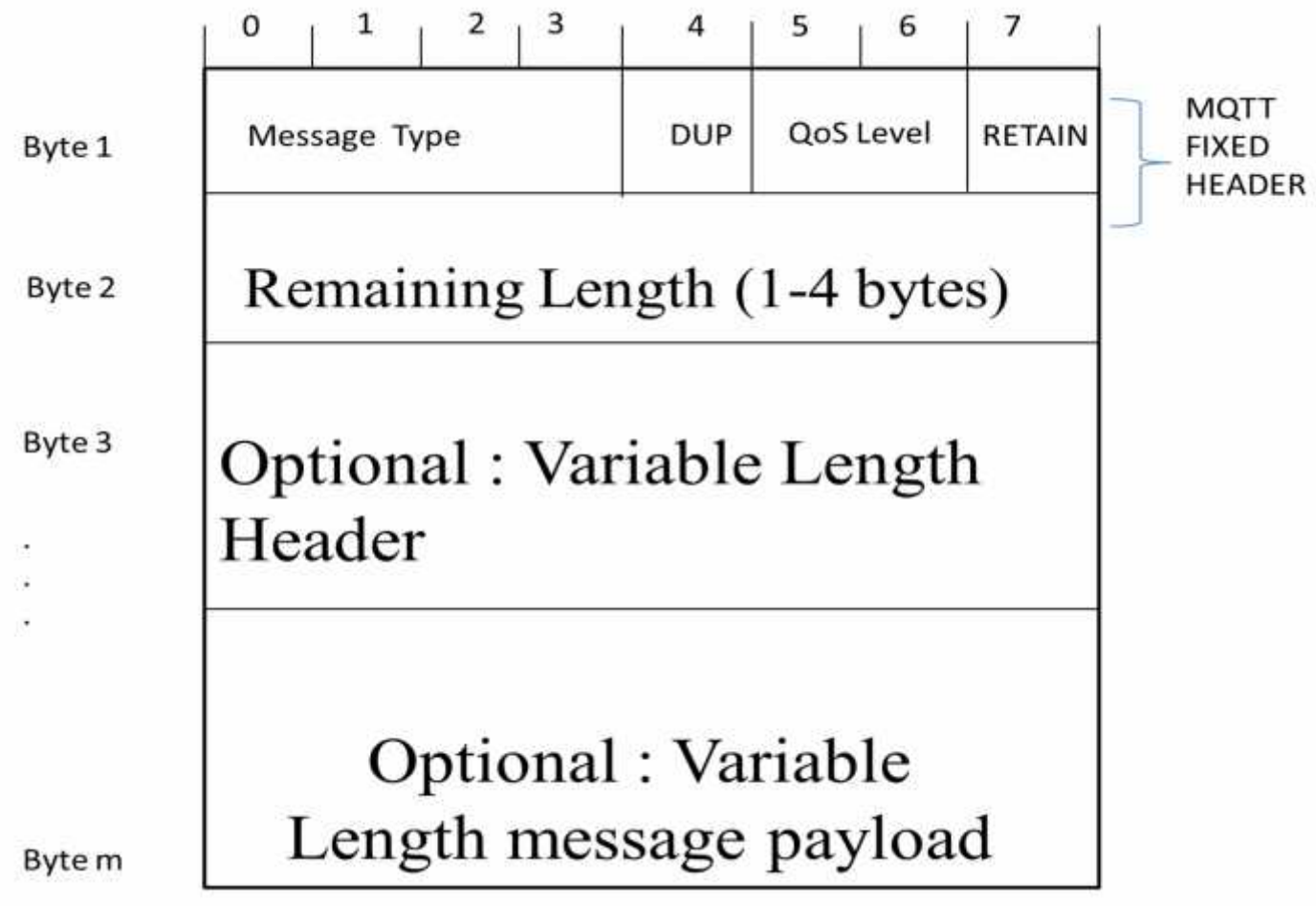

Fig 3.8-MQTT message Format

3.6.2 WebSocket:

The WebSocket protocol makes more interaction between a browser and a website possible, facilitating the real-time data transfer from and to the server. This is made possible by providing a standardized way for the server to send content to the browser without being sought by the client, and allowing messages to be passed back and forth while keeping the connection open. In this way a bi-directional simultaneous conversation can take place between a browser and the server. The communications are done over TCP port number 80, to avoid unnecessary blocking of the connection by the firewall.

3.6.2.1 Web browsers

WebSocket requires web applications on the server to support it and the protocol is currently supported in most major browsers including Google
Chrome, Safari, Firefox. Additionally, WebSocket enables streams of messages on top of TCP. TCP alone deals with streams of bytes with no inherent concept of a message. WebSocket protocol enables two way communications without compromising security assumptions of the web.

The WebSocket protocol specification defines ws and wss as two new uniform resource identifier (URI) schemes that are used for unencrypted and encrypted connections, respectively. Apart from the scheme name and fragment, the rest of the URI components are defined to use URI generic syntax

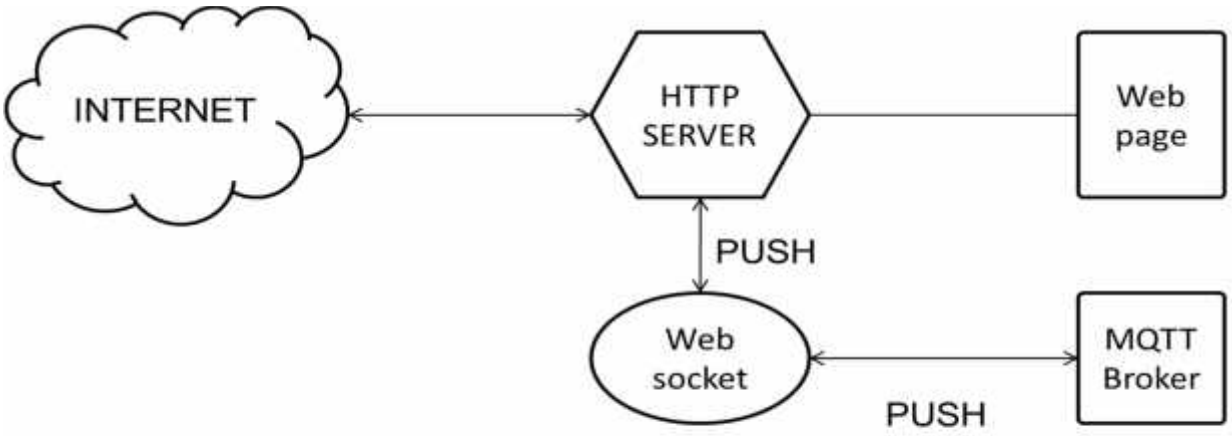

Fig. 3.9-WebSocket interface with HTTP server

3.6.2.2 Initiating WebSocket

To establish a WebSocket connection, the client in this case the MQTT broker sends a WebSocket handshake request, for which the server returns a WebSocket handshake response. The handshake resembles HTTP so that servers can handle HTTP connections as well as WebSocket connections on the same port. Once the connection is established, communication switches to a bidirectional binary protocol that does not conform to the HTTP protocol. Before WebSocket, all communication between web clients and servers relied on HTTP. With the advent of WebSocket, dynamic data can flow freely over WebSocket connections that are persistent, full duplex and fast.

3.6.2.3 Advantage

The main purpose of employing WebSocket is to provide full-duplex communication channel over a single TCP connection that can support any protocol in this case between HTTP and MQTT protocols. While WebSocket was designed to be implemented in both web browsers and web servers, it provides significant architectural benefits to communicate between any client and server that has to connect over the internet, for example in between native mobile applications and servers, and in between devices.

The HTTP server used here supports WebSocket protocol therefore can actively push information to the MQTT broker client any time without the need for polling and the client can push information to the server without doing full HTTP requests each time. The messages are sent on an on-demand basis, thereby drastically reducing the bandwidth usage, which is a boon especially for the Internet of Things. Using WebSocket it is possible to achieve real-time bi-directional data transfer like instant messaging and synchronization of content across browsers.

Generally WebSocket can be used anytime in scenarios beyond serving static Web pages, and require bi-directional communication between the browser and the server. As there is a need for communication amongst two different protocol streams, we use WebSocket as a mediator that forms a channel between both the HTTP server and the MQTT broker.

3.6.3 HTTP

3.6.3.1 What is HTTP?

The web server that is hosting the web page is a generic web server following HTTP protocol. It is an application-level stateless protocol for distributed and collaborative information systems. In this protocol, the browser initiates an HTTP request and after a request is made, the browser disconnects from the server and waits for a response. The server processes the request and re-establishes the connection with the client to send a response back; hence termed as connectionless. It is media independent and can transfer any data format as long as client and server can understand and process the data.

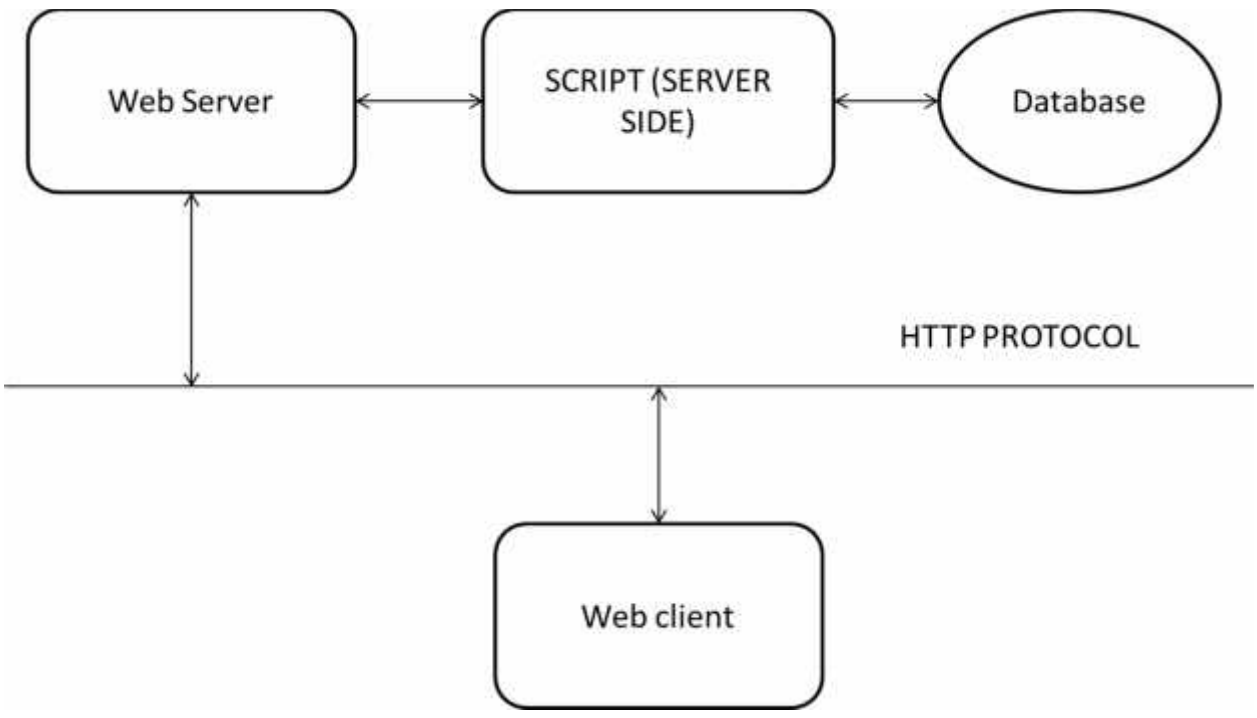

Fig. 3.10- HTTP architecture

Web browsers are made to access HTTP protocol by using the HTML language. In order for a client to receive data from the server, a HTTP request must be sent by the client. Some of the request methods are GET, POST, CONNECT, etc. Similarly the server responds with a HTTP response message. It will contain a status code that determines the type of response from the server, for example, Server error, Client error, etc.

3.6.3.2 Advantages

The reason to employ a HTTP protocol based server is to provide the users with secure connection. A general HTTP server is invulnerable against DNS spoofing and path based attacks. Moreover confidential and sensitive data such as passwords and other details are always stored in encrypted format in the server thereby preventing loss of data; also HTTP is supported by almost all web browsers. Hence HTTP server is an ideal choice to use in this project.

3.7 Programming Languages:

3.7.1 PYTHON:

The programming language Python is an object-oriented, high-level scripting language. It is designed in a way that it is meant to be highly readable. Python is mainly an interpreted language; unlike C language where the source code is first compiled and then executed, python begins its execution line by line without scanning the entire code for error and hence called as interpreter. It has high-level built in data structures, combined with dynamic typing and dynamic binding; make it very versatile for quick Application Development, and also it is used as a scripting language that forms a connection or bridge between the existing components. Owing to its high levels of readability it has simple, easy to learn syntax and therefore reduces the effort taken in program maintenance. Python supports modules and packages, which enable the user to reuse the code at will. The Python interpreter and the extensive standard library are fully open source products and available in source or binary form for major platforms, and can be freely distributed.

Python is gaining recognition among the programming community because it provides increased productivity. As there is no compilation step, the editing, testing, and debugging process is incredibly fast. Debugging Python programs is easy; a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print

statements to the source; the lack of compilation step makes this simple approach very effective.

3.7.1.1 Web server using python

The Flask web framework is written using Python programming language and this is helpful in hosting a web server that runs on HTTP protocol. Flask provides complete functionality over the web server that is being hosted and it has the ability to control the port in which the server is being hosted. In short, Flask provides all functionalities offered by any other generic, well off web services including the ability to serve static files, JavaScripts, PHPs to using WebSockets and HTTP methods.

3.7.1.2 Advantages

The major advantage of using Flask is first off it can be programmed in Python language which is very easy to use. Flask provides an in-built debugger mode functionality that enables user to find any error in their Web Servers. Also it enables user to render a pre-existing HTML template thereby saving time is typing separate HTML files for each user, which is very cumbersome at times.

3.7.2 HTML:

3.7.2.1 Introduction

HTML is a markup language for describing web documents namely web pages. HTML stands for Hyper Text Markup Language. It defines the structure and layout of a Web document by using a variety of tags and attributes. Generally, a markup language is one where a set of markup tags are used to determine the visual

aesthetics of the document, similarly HTML documents are described by HTML tags; each HTML tag describes different document content. These markups are intended for the web browser to interpret and display it in a readable format for the user. It tells the Web browser how to display a Web page's words and images for the user. It ensures the proper formatting of text and images so that your Internet browser may display them as they are intended to look and it can be considered as the bones of a web page. Each individual markup code is referred to as an element or tag. Some elements come in pairs that indicate when some display effect is to begin and when it is to end. There are hundreds of tags that can be used in HTML and all are not supported by every browser, compatibility issues are bound to occur and therefore it is good practice to write the HTML code so that it is compatible with all the browser versions.

3.7.2.2 Advantages

The HTML language, allows for the creation of a perfectly structured web page with proper spacing and aesthetics. The HTML method called “FORM” proves very helpful in creating a secure login page for the users. Also for dynamic communication amongst the client and server the assistance of JavaScript or PHP script mandatory; with HTML there is a facility to write JavaScripts inside a tag called “script” represented as “<script>…. </script>” and this ensures that the code inside that tag is executed as a JavaScript code rather than normal HTML; it is possible to add a pre-existing JavaScript into the HTML file and use the variables from the Script. As HTML provides us with such scalability and agility it is only natural to prefer it over other languages.

3.7.3 JAVASCRIPT:

3.7.3.1 Introduction

JavaScript is an open source, interpreted, scripting language that is developed for enabling programmers to design interactive websites. It is a lightweight programming language equipped with object-oriented capabilities used for creating dynamic environment in the web pages. Owing to its ability to interact with HTML source code, developers have been using it far and wide to make web pages display dynamic contents such as pop-up windows, slideshow of images in the webpages, etc. JavaScript mainly runs on the client side namely in the browser, all browsers do not JavaScript, if a particular browser has its functionality of using JavaScript turned off, the script cannot be run on that browser. It is impossible to use JavaScript to probe into someone else's web site thereby providing better security for the users, moreover even if the script belongs to the same website JavaScript cannot access the server's resources directly like HTML. This is because they are constrained to execute only on the client side namely the browser.

3.7.3.2 Advantages

JavaScript is preferred because of its dynamic nature; it can recognize actions performed by the user such as button-clicks, link navigation and display messages from the user. The main reason to use JavaScript is the availability of library supporting all the MQTT protocol methods. By employing this library the web browser acts as an MQTT client with minimal interaction with the HTTP server. Also the interactivity provided to the client by the JavaScript is exceptional with better interface abilities like drag and drop and slider.

3.8 SOFTWARE:

3.8.1 ARDUINO IDE:

3.8.1.1 Introduction

The open-source Arduino Software (IDE) is used to ease the process of writing code and upload it to any developmental board compatible with this software. It can run on Windows, Mac OS X, and Linux. The Integrated Development Environment is written in Java and the execution is based on Processing projects. It contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the developmental board hardware in order to upload programs and communicate with them.

3.8.1.2 Scripting

The programs that are written in the IDE are generally referred to as sketches; they are written in the text editor region provided in the IDE; like any text editor they support all the common and basic functionalities. The scripts are generally saved with an extension of ".ino". The IDE provides provision for compiling and verifying the code before uploading the code to the hardware of the board that is used. Sketchbook is the concept used by the Arduino IDE. It is a standard place wherein the user can review his most recent scripts and access the same.

3.8.1.3 Uploading

While uploading the code to the hardware it is mandatory to ensure that proper port is selected and baud rate is accepted by the hardware. The data is generally transmitted in a serial fashion. There is a provision for visualizing the transferring

and displaying the data given to the hardware by using Serial Monitor. Depending on the board's hardware proper baud rate must be selected.

3.8.1.4 Language

Most of the boards are supported by the arduio IDE owing to the inputs from the vast community. Each board has its own specific sets of libraries that can be downloaded directly from the IDE. The language that is used in programming the IDE is normal C/C++. Most of the standard C/C++ libraries will function in the IDE with exception of a few; this depends on the hardware specifications of the board connected and the RAM capability.

The NodeMCU can be programmed using the Arduino IDE by downloading a specific set of libraries for the same. The baud rate and the port specification of the IDE is changed in accordance with the hardware specification of NodeMCU and program is uploaded in a serial fashion into the board. The program is written in standard C/C++ language along with the library supporting MQTT functionality. All the standard MQTT protocol methods are supported on NodeMCU.

3.8.2 MQTT BROKER – MOSQUITTO:

3.8.2.1 Introduction

The broker is the most important part of the MQTT protocol; it forms the bridge between all the clients and it is responsible for storing the data regarding the clients like topic, QoS, etc. There are many message brokers that are available to be employed as broker for MQTT, mosquitto is once among them. It is available in the repository and can be downloaded directly or can be built after downloading the repository by using cmake.in case of any Linux based system; it is also available for Windows and Mac OS X.

3.8.2.2 Configuration

There is no specific user interface to run this mosquitto broker; it is executed either as a service or as a command from the terminal. The lack of GUI complicates the process of executing the broker. The mosquitto broker needs special WebSockets library in order for it to support WebSocket protocol. Like any normal web server the MQTT broker also needs an IP address to operate and function over Internet.

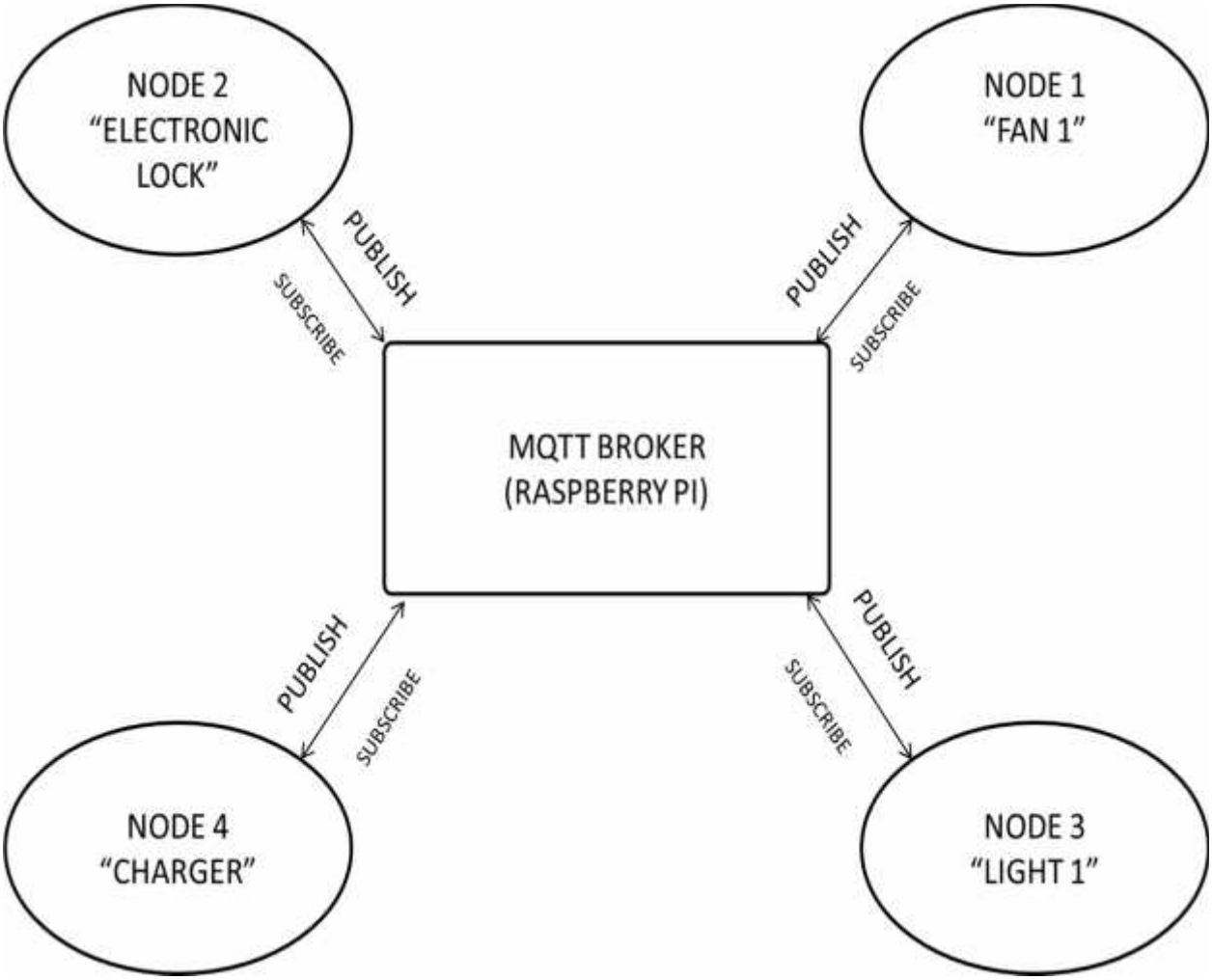

Fig 3.11-Simple broker and client Interface

The IP address and the port over which the MQTT protocol communication occurs are specified by the user inside the configuration file used by the mosquitto broker on start up. The mosquitto broker must be configured carefully so that proper port is assigned for MQTT protocol and it is capable of supporting WebSockets.

3.8.2.3 Advantage

The mosquitto broker hosted on the Raspberry Pi is wirelessly connected to four nodes, each connected to its own disparate appliance. It is the duty of the broker to keep track of the number of nodes, the total number of topics, and their subscriptions and publishes. Here all the nodes are subscribed to each other's topic, so if Node 3 publishes that "light is ON" all the nodes in this network will receive that message as a result of their subscription. The broker ensures that the message delivery is instantaneous and omnipresence.

3.9 PORT FORWARDING:

3.9.1 Introduction

Port forwarding is more of an executive step to be performed rather than describing it as software. Port forwarding is the process of redirecting a communication request from a specific address and port number combination to another while the packets are traversing a network gateway, such as router. This is due to the advent of Network address Translation. All the IP address are assigned dynamically using the DHCP protocol, basically the IP address of your device changes repeatedly while using Internet. The web Server that we host is a local network namely LAN, so in order for it to be accessed from anywhere in the world we need to employ port forwarding.

3.9.2 Implementing port forwarding

The IP address of the device hosting the web server in the LAN (also called as private IP) is bound to its IP address used for accessing the Internet. Therefore when a HTTP request is performed to the public IP address of the server it is port forwarded by the router to the private IP address of the server, thereby forming a channel to the Internet. The same process is executed for hosting the msoquitto broker over Internet.

4 Conclusion

The appliances are triggered and controlled by the user via the web page. The status of each and every one of those interconnected appliances is monitored with ease. The transmission of data is swift, instantaneous, and mainly transmitted to every client connected to the Internet. Depending on the environment in which the client is deployed, the quality of service of the MQTT protocol can be varied to obtain the desired result. MQTT being a light weight protocol and also it works as an overhead to the pre-existing TCP/IP protocol, so there is no need to ponder over the question of compatibility as most of the Internet connections are based on TCP/IP protocol.

5. FUTURE WORKS:

Though this methodology is unique and fairly new in origin, there is always scope for further improvement. Some of them are

1. The power consumed by the appliances can be calculated by keeping track of the time duration the appliance is ON and knowing its power rating
2. A database can be developed on when the appliances switches ON and OFF with details like date, time, which user accessed the device, etc.
3. Instead of having a webpage that is accessed by button clicks, we can deploy a voice recognition system and use the output of that system to trigger appliances.

There are endless possibilities for developing a system with better functionality and making it more robust, efficient and secure.

REFERENCE:

1 Xue Li ; Sch. of Comput. Sci. & Technol., Harbin Inst. of Technol., Harbin, China ; Lanshun Nie ; Shuo Chen ; Dechen Zhan, An IoT Service Framework for Smart Home: Case Study on HEM,Mobile Services (MS), IEEE International Conference, June 27 2015 Pg. 438 - 445

2 Asghar, M.H. ; Sch. of Comput. Sci. & Inf. Technol., Univ. of Hyderabad, Hyderabad, India ; Mohammadzadeh, N., Design and simulation of energy efficiency in node based on MQTT protocol in Internet of Things , Green Computing and Internet of Things (ICGCIoT), 2015 International Conference, 8-10 Oct. 2015: 1413 - 1417

3 Seong-Min Kim ; Dept. of Multimedia Eng., Hanbat Nat. Univ., Daejeon, South Korea ; Hoan-Suk Choi ; Woo-Seop Rhee, IoT home gateway for auto-configuration and management of MQTT devices, Wireless Sensors (ICWiSe), 2015 IEEE Conference, 24-26 Aug. 2015: 12 - 17

4 Singh, M. ; TCS Innovation Labs., Bangalore, India ; Rajan, M.A. ; Shivraj, V.L. ; Balamuralidhar, P., Secure MQTT for Internet of Things (IoT), Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference, 4-6 April 2015 : 746 - 751