AMITY UNIVERSITY

UTTAR PRADESH

In-House Project Report

on

EXPLORING IOT HOME AUTOMATION USING ESP8266

Submitted to

AMITY UNIVERSITY UTTAR PRADESH

In partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

In

Computer Science & Engineering – 3C

By

SAHIL MAKWANE

A2372016030

Under the guidance of

MS. SHANU SHARMA

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY

AMITY UNIVERSITY UTTAR PRADESH NOIDA (U.P.)

ABSTRACT

Internet of Things home automation industry is opening the flood gates to a whole new world of sensors and microcontrollers that reduce human effort and simplify how usual things function around us. Things like the thermostat, lights, wireless monitors, etc., are being innovated for us to be able to control such commodities using just a mobile device. Compact chips/modules are constantly being developed and worked upon in order to bridge the gap between technology and idea of IoT. One such module, ESP8266, which came out a few years back storing a ton of potential, has a lot left to be explored. Being a programmable module having Wifi protocols, ESP8266 has brought in new opportunities like never before.

This project aims to explore ESP8266 and its capabilities from the basics. By establishing a Web Server, we explore the ability of this module to use wifi protocols, being a microcontroller. The setup process for the apparatus requires precision and perseverance. Once accomplished we move on to the equally difficult process of setting up an Integrated Development Environment (IDE) and pushing our program for the server onto the module.

Keeping in mind the potential of the module, the future scope including applications using the module has been discussed in this project.

Table of Contents

List of Figures

CHAPTER I: INTRODUCTION

1.1 Introduction

Internet of Things, popularly known as IoT is a wireless-system of everyday devices, the objects we see around us, lights, machines. Each object, is assigned a Unique Identifiers (UIDs) and has the ability to transmit data over to the network without an interference from people. Like, controlling the water sprinkler wirelessly from our bedroom by using just our phone; as simple as pressing a button can switch on the sprinkler in our garden, remotely. IoT involves extending internet linking through devices which we wouldn't usually link with computing. Having interlinked with technology these devices can be remotely monitored and controlled.

The concept of IoT dates back to the 80s wherein a modified coke dispensing machine became the first Internet-connected appliance at Carnegie Mellon University. Overtime, the definition and understanding of IoT has evolved due to convergence of multiple technologies, real-time analytics, machine learning, sensors, embedded systems, microcontrollers etc. As of today, there are hundreds of different devices that allow us to remotely extend internet to things and control them. One of the most popular devices that caused a disruption in the industry was Raspberry Pi but it came under the category of a computer with a hefty power. Then in 2014, came **ESP8266**.

1.2 Motivation

The current hype of IoT lies in the sensor-based world where everything would be controlled by devices remotely. Applications go as broad as one can think of while implementation is still progressing as technology advances. However, intriguing it may sound to control everyday objects using your mobile phone, it is imperative to understand everything from ground-up to actually implement something of the sort.

ESP8266 is a relatively new chip with limited documentation and a small community of developers. One of the developers happened to be a friend of my dad's. He introduced me to ESP8266 chip a few months back and it was a meeting I will remember for a long time.

Not only did I not understand half of what he was trying to teach, I thought of it all as boring. Towards the end of the meeting wherein he concluded by stating that this chip is the absolute basic of how it all begins for the domain of IoT, it all made sense. This last statement of his actually intrigued me to the point where I did my own research, following which I was hooked onto this project. Having no electronics background, I started my research with the aim to understand how IoT domain functions at the grassroot level.

1.3 Objective

To build a web server using ESP8266 microcontroller. A web server is one of the most basic implementations using the ESP8266 chip, but is not nearly as simple as it may sound.

The following are the key objectives (in no particular order):

- A proper functioning circuit with connection for the ESP chip.
- Flashing the module with the correct interpreter of LUA
- Uploading web server code to the ESP module using ESPlorer
- Establish web server and controlling an LED using mobile

The final project will be an implementation to demonstrate the working of IoT.

1.4 Report Structure

This report is organized in the following chapters:

Chapter 1: Introduction to IoT, Motivation, Objective, Report Structure.

Chapter 2: Explanation of the different components used along with the hardware apparatus.

Chapter 3: Running Web Server using the apparatus and making sure there are no errors.

Chapter 4: Result of the overall project along with the pictures of the demonstration.

Chapter 5: Future prospects of the module along with the conclusions established.

CHAPTER II: MATERIAL & SET-UP

2.1 ESP8266

The **ESP8266** is a microcontroller (A microcontroller is an integrated circuit that is capable of running application program but unlike a PC, these are very low-end with a really limited amount of memory and storage capacity) as well as a low-cost wifi microchip (Has the capability to access internet and provide wifi) Initially produced by Espressif Systems (a Chinese company), the ESP-01 module came to western developers in 2014 when 3rd party companies started manufacturing it. Advertised as a wifi solution, this microchip merges the capabilities of a microcontroller to that of wifi. The ESP8266 has in its reach the means to sustain self-contained applications. The ESP8266 can work as both, an Access Point as well as a Station having wifi capability.

The extremely low price point (Rs.200 for the one I purchased) and the limitation of external components has made it possible for this chip to be of this low a cost and has even piqued the interest of a lot of developers around the globe. A whole community formed by ESP module developers has helped develop many software and document the on-going progress for various applications involving work in several domains.

The module that this project focuses on involves the ESP8266 - 01 module developed by Ai-Thinker.

Fig.2.1 ESP-01 (Front & Back)

Measuring just 14.3 x 24.8 mm in dimension, the ESP-01 comes in two variants; one with 512kb of Flash Memory and the other with 1Mb of Flash Memory (The one I have used is the 1MB module). ESP-01 has 2 GPIO (General Purpose Input/Output) pins based off of a L106 32-bit RISC microprocessor core.

Note: RISC (Reduced Instruction Set Computer) is one whose instruction set architecture allows it to have fewer cycles per instruction.

2.2 ESP8266 Pinout

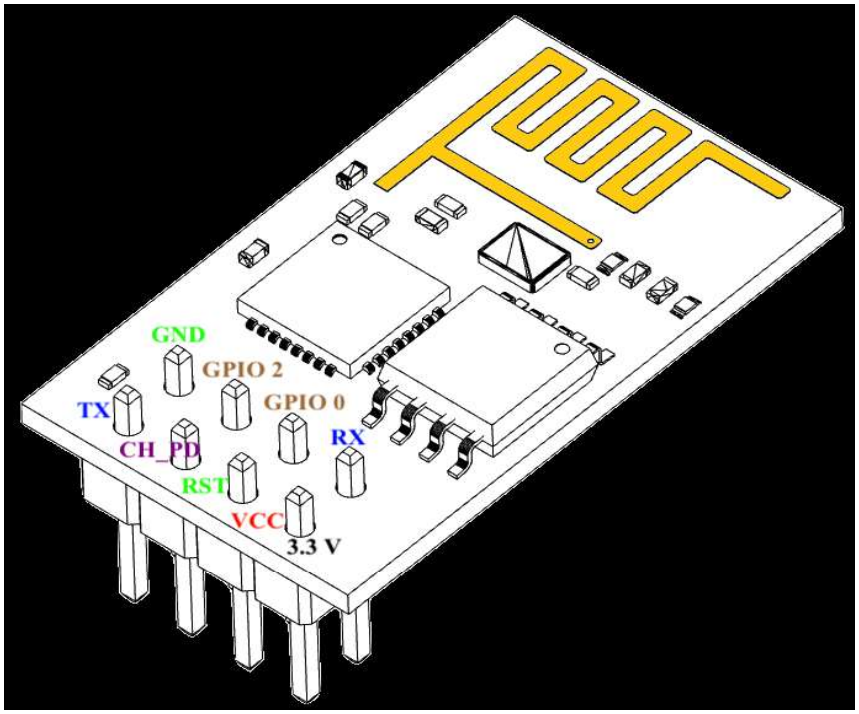

Fig.1.2 ESP-01 Pin Layout

The pin-layout for the ESP-01 is as follows:

- GND, Ground (0 V)
- VCC, Voltage (+3.3 V; can handle up to 3.6 V)
- RX, Receive data bit X
- TX, Transmit data bit X
- CH_PD, Chip power-down
- RST, Reset
- GPIO 0, General-purpose input/output No. 0
- GPIO 2, General-purpose input/output No. 2

Raspberry Pi's are often used by developers and students for development of applications as well as Internet of Things projects. They provide much more memory and a micro-SD card storage. However, a Raspberry Pi falls under the category of a computer, not a microcontroller giving edge to the ESP modules at what they do.

The ESP8266 modules have an explicably good capability to direct electrical inputs and outputs (GPIOs). Thus, combining the capabilities of microcontroller with native wireless internet giving us a processor that could run applications as well as or better than an Arduino, which would have hardware support, RAM and flash memory but would also have the unique ability of forming Internet connections. Simply put, this is what ESP8266 module is.

2.3 Basic model of Home Automation

The way in which the basic home automation using IoT works is shown in the following diagram:

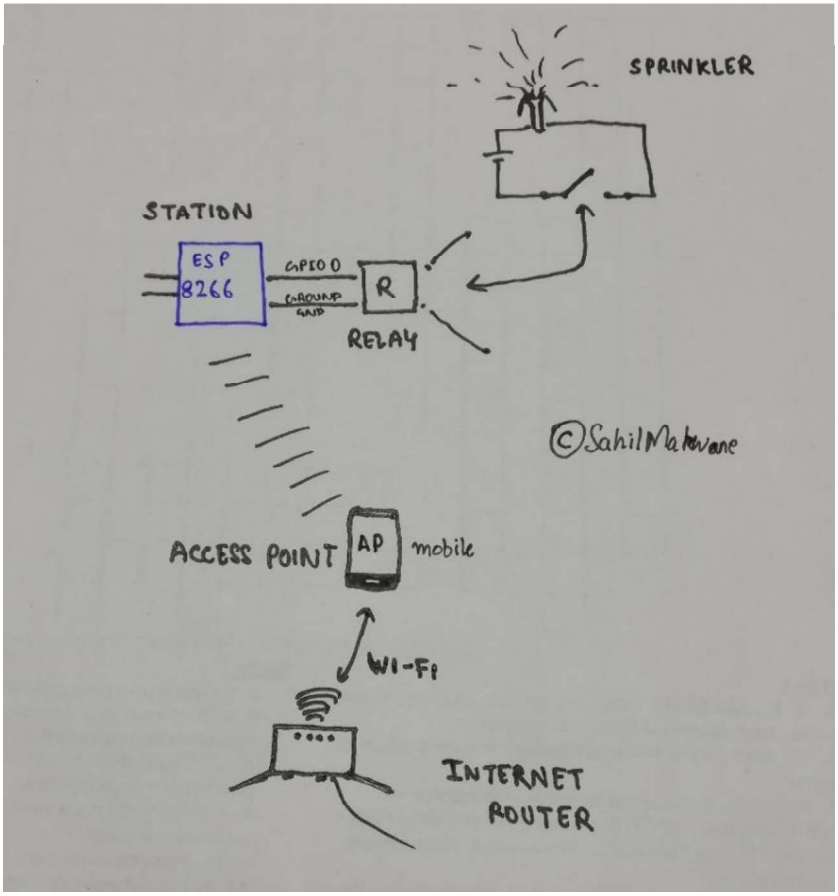

Fig.2.3 Basic functioning of Home Automation

The above diagram features a sprinkler connected to a circuit whose switch is controlled by a Relay. The Relay is connected to ESP8266 which is connected wireless to an Access Point (mobile) through wifi provided by the home internet router.

***Relay**: A relay is an electrically operated switch.*

***ESP8266**: It a microcontroller with wifi capability. This module is acting as a **Station**. It receives a request from the **Access Point** (which can be our mobile or any other device). Station and Access Point are discussed later in the project.*

***Internet Router**: The simple router you have at your house that is providing internet.*

Home automation as we know it has advanced quite a lot in the recent years but to understand anything, we have to begin with the basics. The above diagram is as basic it gets in the world of home automation using IoT. With the blooming Internet of Things industry with prime attention to automation and wireless connectivity, small yet powerful devices having several capabilities are being put to development. ESP8266, a microcontroller with wifi abilities, even smaller in size than most of the currently known processors, was welcomed interestingly by the developer community in the IoT domain.

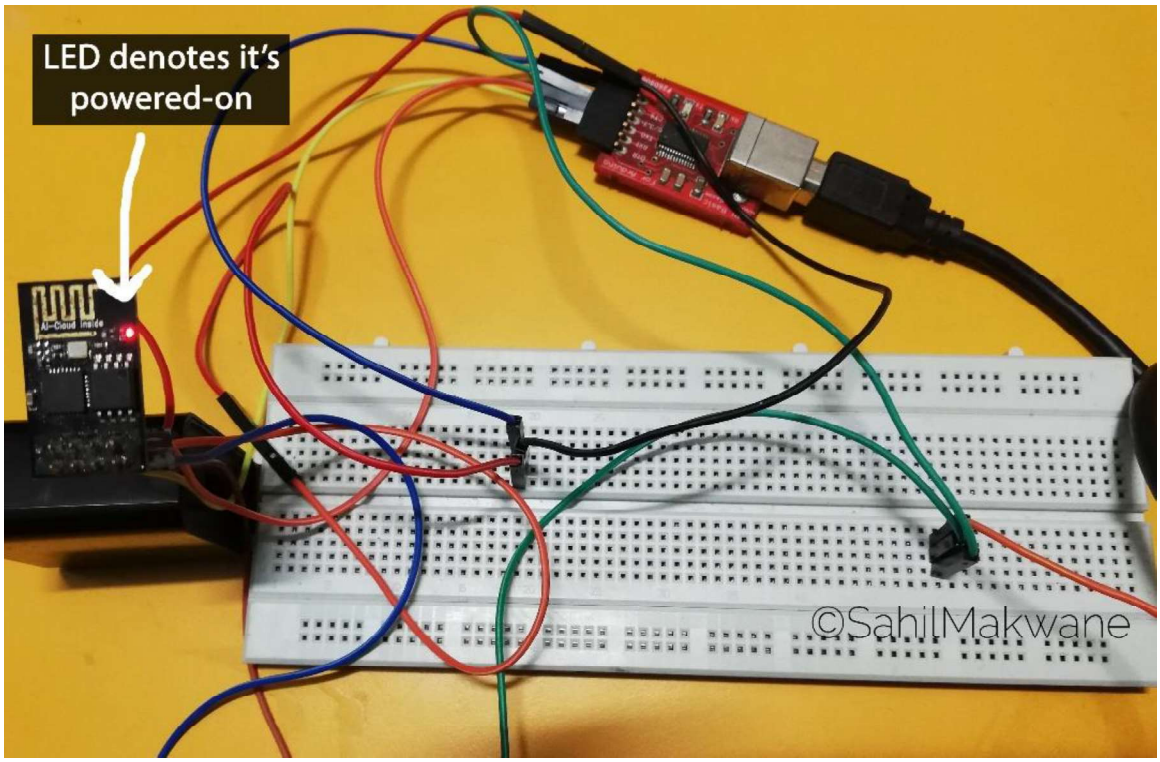

Fig.2.4 Basic Layout

2.4 Hardware required

The required hardware parts are:

- ESP8266-01 (or simply ESP-01) module
- FTDI Programmer
- 1x LED
- Breadboard
- Jumper Cables

We cannot program an ESP8266-01 without supplying it data through a UART (Universal Asynchronous Receiver-Transmitter whose purpose is to transmit and receive serial data). ESP-01 cannot be connected directly to the computer. The easiest way to achieve this is through the use of a USB-UART converter. A popular brand are the devices from Future Technology Devices International (FTDI) are used for the same. An FTDI programmer is used for this project.

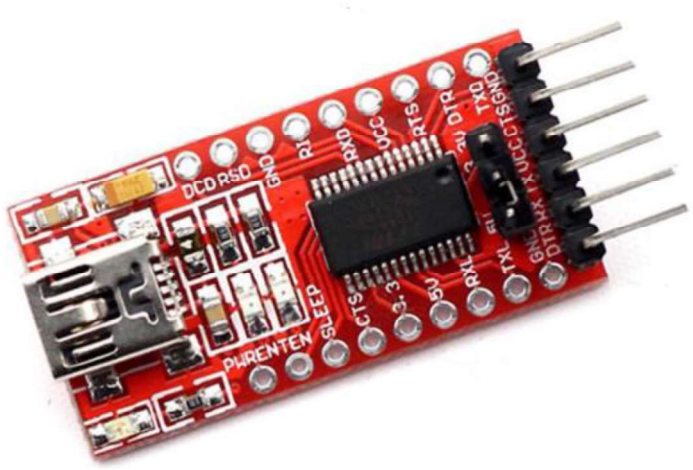

Fig.2.5 FTDI Programmer

A serial-communication of ESP module with FTDI is needed to enable us to upload code to the ESP-01. The figure for the same is shown below.

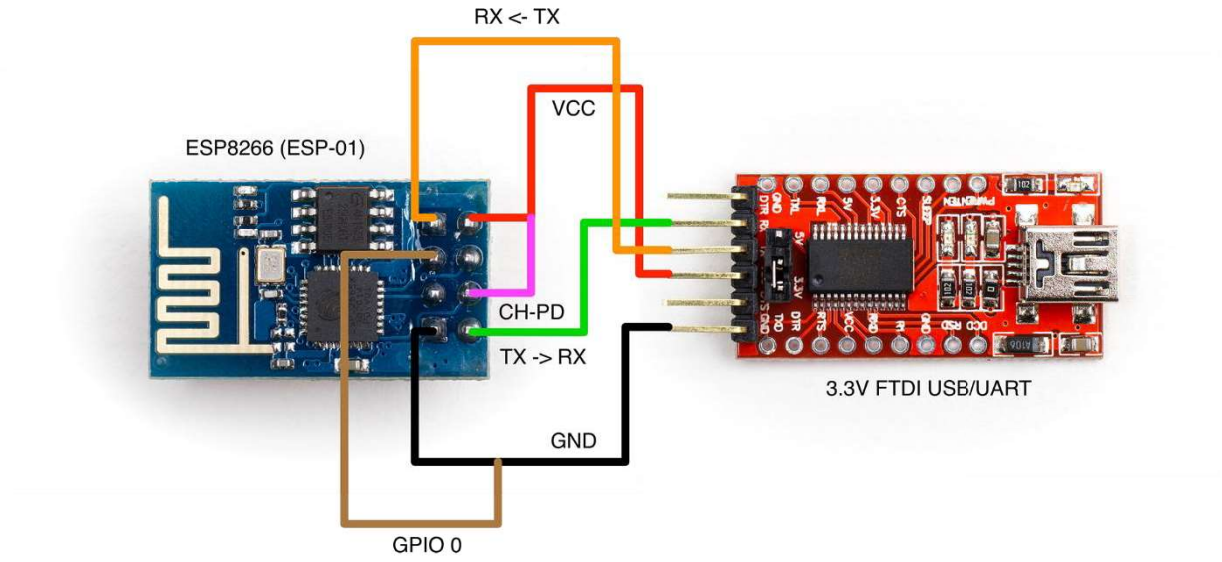

Fig.2.6 Connection of ESP-01 module with FTDI Programmer

The FTDI is the one which gets connected to the port of the computer. The connection of ESP with FTDI in terms of pins-on-board is tabled below.

ESP8266	FTDI programmer
RX	TX
TX	RX
CH_PD	3.3V
GPIO 0	GND
VCC	3.3V
GND	GND

Fig.2.7 ESP – FDTI pins connection

CHAPTER III: SIMULATION & EXPERIMENTATION

The ESP8266 is a wifi device having the ability to connect using some protocols but some assistance is required. The network parameters (network name, password, etc.) are not known to the module. This is when we are connecting it as a station. To make the device an access point we can use the UART. The TX pin enables outbound transmission while the RX pin enables the inbound transmission of data. A secondary usage of the UART is to receive binary data used to flash the flash memory in order to record new programs for running.

3.1 Wi-Fi Theory

While working with microcontrollers like ESP-01 which is wifi oriented, we need to understand some wifi concepts that are integral for basic understanding of what works how. At high level, Wifi is the ability to participate in TCP/IP protocols over a wireless communication link.

A device called Wireless Access Point acts as the hub of all communications. It is connected as TCP/IP router to the rest of the TCP/IP network. WiFi connections are then formed to the access point (through devices called stations) and TCP/IP traffic flows through the access point to the Internet. The devices that connect to the access points are called "stations".

An ESP8266 device can play the role of an Access Point, a Station or both at the same time. Very commonly, the access point also has a network connection to the Internet and acts as a bridge between the wireless network and the broader TCP/IP network that is the Internet.

3.2 Flashing ESP module with NodeMCU

NodeMCU is a firmware which enables users to code application programs into the ESP8266 Module using LUA script. A wifi can be setup, GPIOs can be controlled, ESP8266 modules can be turned to a server and more.

LUA is a scripting language that is available on ESP8266 environments. The environment used for this project is the ESPlorer Integrated Development Environment (IDE). The most popular implementation of Lua for the ESP8266 is known as the NodeMCU Lua firmware and is available at its github repository. Builds of the firmware can be downloaded directly as well as the source. Once you have a copy of the firmware you can flash it using any flash tool.

Flashing means embedding the ESP chip with an interpreter for the program/code that we push to it using the IDE. In this project, the ESP chip is being flashed with a binary file (.bin file) that can interpret the LUA code we upload onto it. NodeMCU flasher is being used here while any flashing tool can be used. Opening the flasher exe opens the following window:

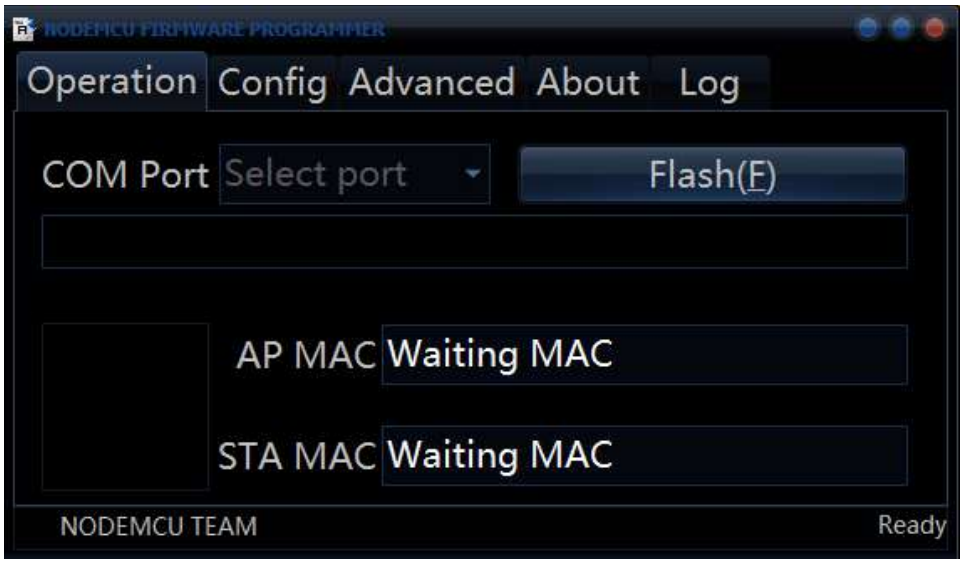

Fig.3.1 Flasher default window

Make sure that GPIO 0 is grounded. Select the port to which the module has been connected. Click the Flash(F) button to start flashing. Once flashing is done, a green circle

should appear on the bottom left denoting that flashing was a success. The following image shows the same:

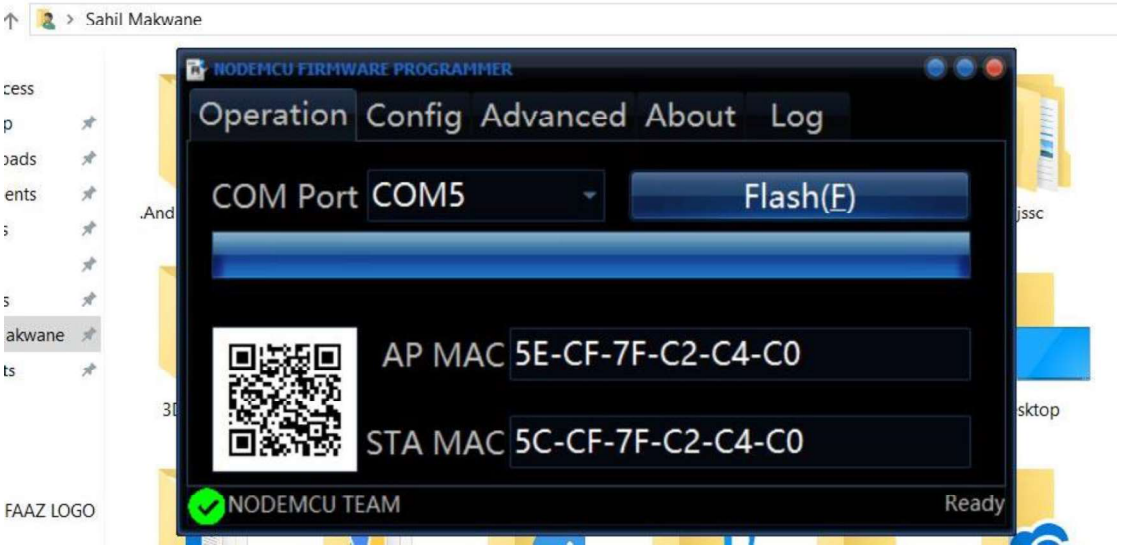

Fig.3.2 Finished Flashing

3.3 Sending code to the module

1. Remove the module from the USB port in the computer and un-ground the GPIO 0.
2. Open ESPlorer folder.
3. Run ESPlorer.jar (which is an Executable Jar File)

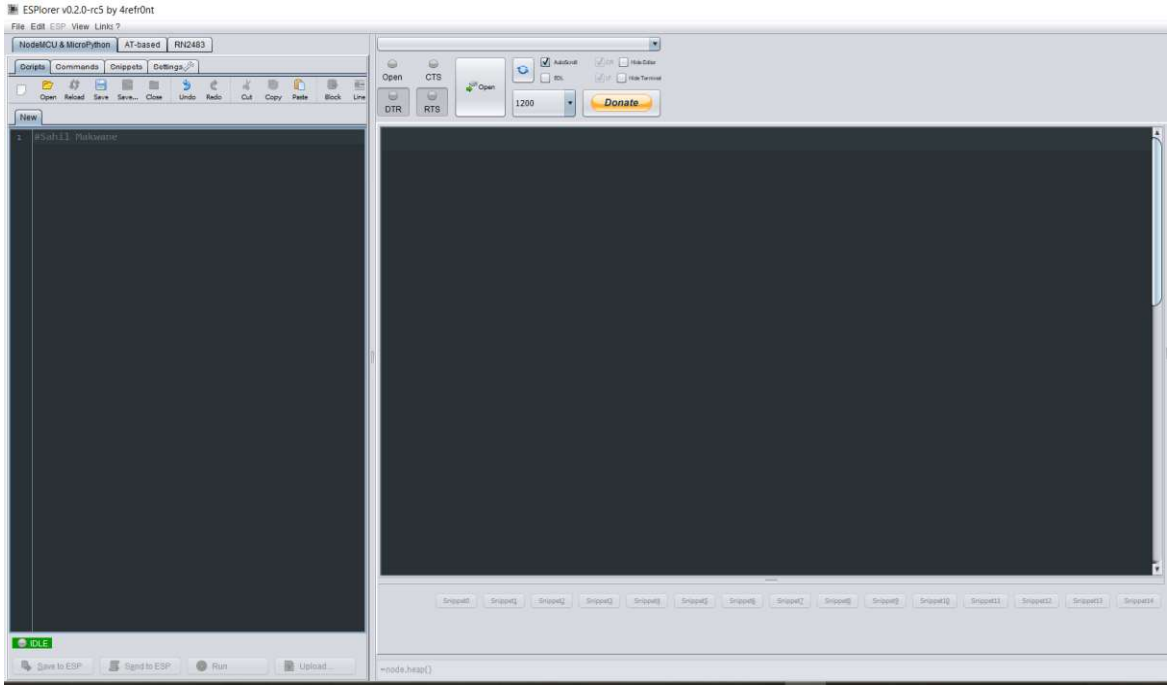

Fig.3.3 ESPlorer IDE

4. Connect the module back into the USB port of the computer.
5. Press Open button on the top right.
6. Make sure NodeMCU+MicroPython tab is active on the left column and create a new file called init.lua.
7. Save it to ESP.
8. Upload the following code to init.lua:

Note: The code is written in Lua. Lua is a lightweight, cross-platform programming language designed in ANSI C, primarily for embedded use in applications.

Understanding each line of code in depth is a redundancy for now. The main focus is on the application at the beginner level. We shall look more in-depth of the code once we familiarize with the functionality of ESP8266 as well as ESPlorer.

```
1  -- this is the preparation of pin 3 of esp8266 to be made as output.
2  gpio.mode(3,gpio.OUTPUT);
3  wifi.setmode(wifi.STATIONAP)
4  wifi.sta.config("sahil","abcd1234")
5  print(wifi.sta.getip())
6
7  -- this is the web server section of code
8  srv = net.createServer(net.TCP)
9
10 local buf="";local rs=0; srv:listen( 80,function(conn)
11 conn:on("receive",function(client,request) rs=string.len(request);
12 buf="<p>GPIO0<a
13 href=\"?pin=ON1\"><button>ON</button>";buf=buf.."</a> </a></p>";
14 buf=buf..tostring(rs);   if(rs >= 323)then gpio.write(3,gpio.HIGH);
15 tmr.delay(100000); gpio.write(3,gpio.LOW); buf=buf..tostring("<--
16 thanks..the relay just tripped"); end;client:send(buf);
17 client:close();
18 end)
19 end)
```

Fig.3.4 Web Server Code

9. Replace your WiFi Station details in the code above at line 4 (Network Name and Password).

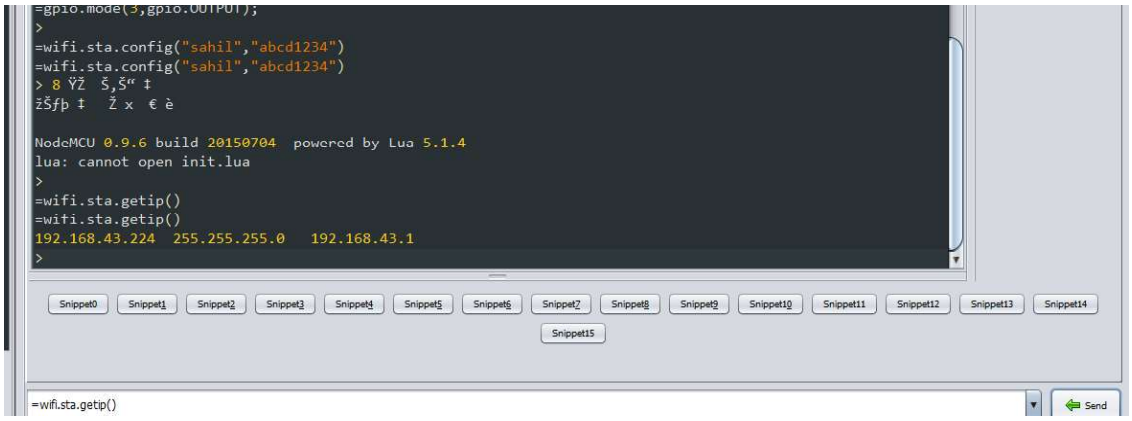

Fig.3.5 IP Addresses

Note: The IP addresses (at the last line) in the figure above represent the IP of ESP chip, mask IP, IP of the wifi, respectively.

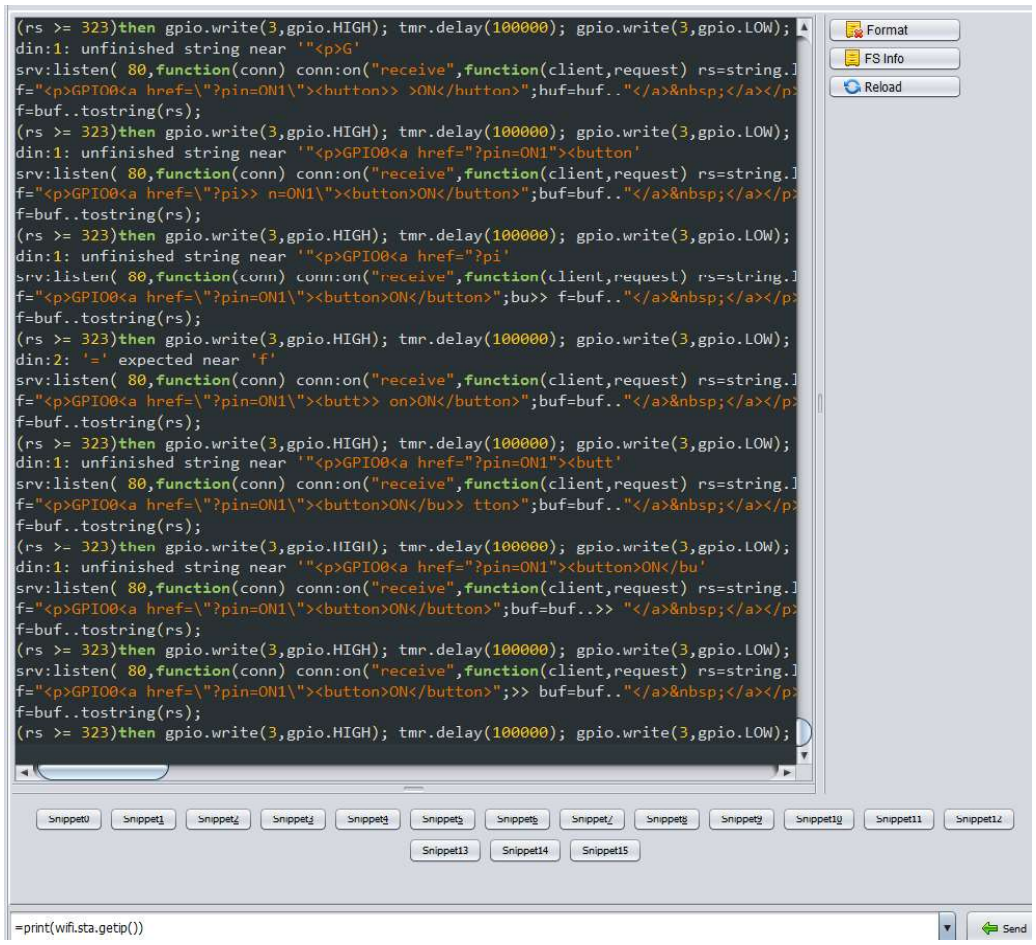

Fig.3.6 Buggy nature of ESP with the code finally compiling in the last line

Note: The above figure shows the redundant errors shown by the ESPlorer compiler. The primary thing to note here is that fact that these aren't legitimate error. One has to compile the same piece of code a lot of times before it finally works (as it did in the last line of the above). Regardless of errors in the code, the ESPlorer shows errors (which is a bug).

CHAPTER IV: RESULTS & DISCUSSIONS

4.1 Accessing the Web Server

Once you have the IP address of the ESP chip. Select the ESP wifi in your WLAN options in the notification bar.

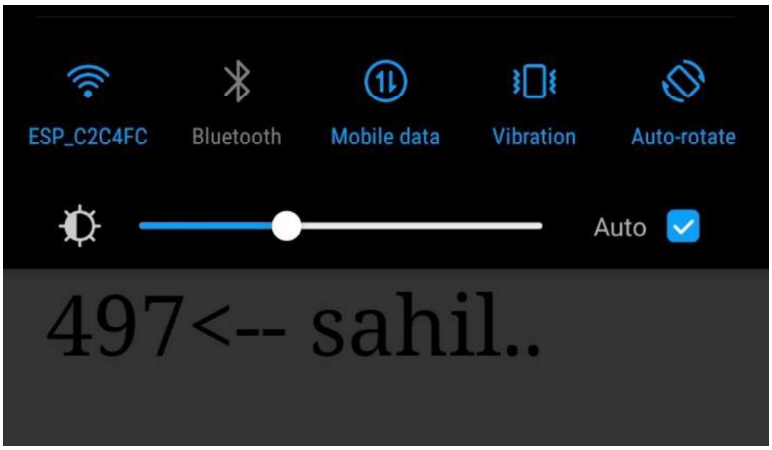

Fig.4.1 The ESP chip is acting as a station and providing the Wi-Fi

The IP address is accessed on the mobile browser. An ‘On’ button appears along with a number and “sahil”. The number here represents the length of the request string that is sent to the ESP. Pressing the On button controls the LED as well as the number increments (as we’re sending more requests).

Fig.4.2 a: Web Server b: Pressing On button changes URL and increments the number

Fig.4.3 The new URL

Fig.4.4 Pressed On once more

Fig.4.5 a: LED ON b: LED OFF

Note: The LED is actually On while we press the button and as soon as we press, it shuts off and switches back On in fractions of a second. The light basically blinks on pressing the button. The above images are stills from a video that was recorded of the same.

CHAPTER V: CONCLUSION & FUTURE SCOPE

5.1 Conclusions

We have finally established the Web Server that we intended to. The process (as I have defined it) is relatively simple and to the point. The time, effort and hard-work it took to reach this stage, however, was spread over months.

Having concluded the project of building a web server using ESP8266 module does not conclude my interest in the domain of Internet of Things. This project marks the beginning of my indulgent into new developments using the ESP8266 module. The fact that this microcontroller is unknown to a lot of the developers (as it is relatively new) gives me the motivation to work on paths that are yet to be explored.

One major thing on which light needs to be shed upon, is that the ESP8266 module is very hard to work with. Setting up the whole circuit using all the hardware takes hours to setup properly and diagnose (if problems persist). Working with ESPlorer is a hardship in itself. Even though there are stable versions for the IDE, it is full of bugs. Here are some common problems that I faced during the project:

- Wind, dirt & noise affect the working of the ESP module. However unfounded it may sound; the module requires a clean and quiet environment to work efficiently.
- Jumper cables are not reliable and need to be checked on a regular basis (sometimes after every 2 mins).
- UART to USB converter cable varies for different models of the FTDI programmer. One of my FTDIs did not work due to unmatched cable.
- ESPlorer is way too buggy to work with patiently.
- ESPlorer did not even load on my Windows 10 while using the latest version of Java, Java 10. After an hour of searching for a remedy, I fell upon a suggestion to revert back the version of Java. It turns out ESPlorer supports Java 8 on Windows 10 and not newer versions.

- When moving the apparatus from one place to another, **do not** disassemble it. It takes as much time setting it up as it took while setting it up the first time [personal experience].

5.2 Future Prospects

Being a relatively new module in the realm of IoT, coming into existence in 2014, ESP8266 has plethora of un-explored features, functions, capabilities, waiting to be documented. The module is yet to be accompanied with manual, proper documentation or even informatic tutorials. The usage and working of the ESP8266 are still being worked out by developers and IoT enthusiasts around the world.

As we go ahead with the enormous advancement in technology on a daily basis, the IoT sector is concentrating more towards the automation sector attracting more consumers as it grows. Ease of use and 'easily-reachable' is something which the majority desires on this date. Catering to the same, home automation strikes a home run with what it provides. Compactness along with power makes a device get attention and enable developers to explore it. The story was the same for Raspberry Pi as it was introduced to this world. ESP8266 is the next Raspberry Pi, potentially better. Although in terms of size, what a Raspberry Pi is to a small PC, an ESP8266 is to a Raspberry Pi. Although small, but full of power.

Having already discussed the capabilities of this module which bridges the gap between microcontrollers and wifi protocols, one shall wonder the functionality of such a device in contributing the IoT home automation domain.

ESP8266 application subjects include (but not limited to):

- Smart Power Plug
- Home Automation
- Mesh Network
- Industrial Wireless Control

- Baby Monitor
- Network Camera
- Sensor Networks
- Wearable Electronics
- Wireless location-aware devices
- Security ID Tag
- Wireless Positioning System Signals

If one wants to follow the routes that have been taken before, there are a handful of microcontrollers and processors (like Raspberry Pi, Arduino, etc) which would appear more captivating. But, if one wishes to explore the unknown and start from level-0 of newly arrived devices, ESP8266 is the way to go.

REFERENCES

[1] Rui Santos, “Getting Started with the ESPlorer IDE” [online]
Available: https://www.scribd.com/document/331424634/Getting-Started-with-the-ESPlorer-IDE-Rui-Santos-pdf

[2] “ESP8266EX Datasheet” [online]
Available: https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266__Datasheet__EN_v4.3.pdf

[3] “Kolban’s book on the ESP8266” [online]
Available: http://neilkolban.com/tech/esp8266/

[4] Build an ESP8266 Web Server – Code and Schematics” [online]
Available: https://randomnerdtutorials.com/esp8266-web-server/

[5] “ESP8266 module comparison: ESP-01, ESP-05, ESP-12, ESP-201, Test Board and NodeMCU” [online]
Available: https://blog.squix.org/2015/03/esp8266-module-comparison-esp-01-esp-05.html