Pocket Server Using Raspberry Pi firewall

by

Rachit Bhanage

Submitted in partial fulfilment of the requirements for the degree of

MASTER OF ENGINEERING

Major Subject: Internetworking

at

DALHOUSIE UNIVERSITY
Halifax, Nova Scotia
April, 2019

Dalhousie University

Faculty of Engineering

Internetworking

The undersigned hereby certify that they have read and award a pass in INWK 6800 for the course/seminar entitled "Pocket Server Using Raspberry Pi firewall" by *Rachit Bhanage* in partial fulfilment of the requirements for the degree of Master of Engineering.

Bill Robertson

DALHOUSIE UNIVERSITY

INTERNETWORKING PROGRAM

AUTHORITY TO DISTRIBUTE REPORT

Title:

Pocket Server Using Raspberry Pi Firewall

The Internetworking Program may make available or authorise others to make available individual photo/microfilm or soft copies of this report without restrictions after 2020

The author attests that permission has been obtained for the use of any copyrighted material appearing in this report (other than brief excerpts requiring only proper acknowledgement in scholarly writing) and that all such use is clearly acknowledged.

Full Name of Author: Rachit Bhanage

Signature of Author: ___________________________

Date: ___________________________

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

EMMC	Embedded multi-media controller
GHZ	Gigahertz
HTML	Hypertext markup language
IP	Internet Protocol
IOT	Internet of things
LAN	Local area network
LED	Light-emitting diode
OS	Operating system
USD	United States dollar
RAM	Random access memory
SD	Secure digital
SDK	Software development kit
SOHO	Small office home office
SSH	Secure shell
SSID	Service set identifier
UFW	Uncomplicated firewall
USB	Universal serial bus
WIFI	Wireless fidelity

EXECUTIVE SUMMARY

Technology giants like Cisco, spends approximately **$6332 million USD (2018)** on Research & experimentation as evident from the below bar-graph (*Figure1.1*) [Sta2019]. For small scale industries or SOHOs cannot afford such huge financial investments.

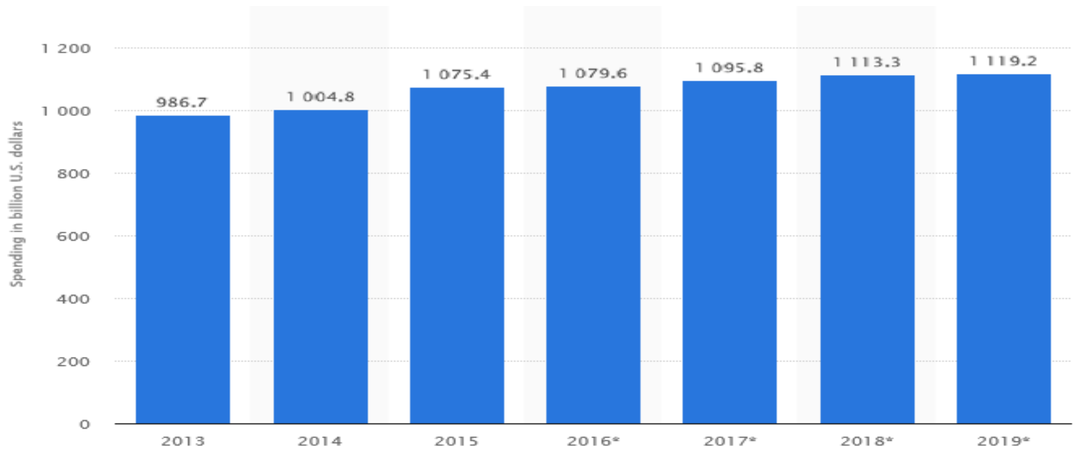

Figure1.1 Cisco's expenditure on research and development - 2017. Amount in $Million USD [Sta2019].

If we are able to come up with a cheap testing hardware and an open-source/cost-effective software to compliment it, we can effectively reduce the cost with a huge margin. To address the above problem, we propose the following:

- Make use of emerging technologies like Internet of Things (IOTs) [Nei2004]. Much research is to derive from them.
- IOTs are know for their cheap & inexpensive hardware.
- Use open-source software and tools wherever possible to leverage the cost-effectiveness and also contributing the community.

To convert the entire infrastructure & research environment, a tight integration will be required among the various different communicating devices as there is a lot of diversity and no proper standardization. This is the risk factor and a challenge to overcome. While most of the system can be developed utilizing existing infrastructure, some minor initial investments might be required.

1 INTRODUCTION

Internet of things can be utilized for an inexpensive substitution for the current technological challenges. They can be bifurcated into: Industrial IOTs & Domestic IOTs. For the scope of the project we are going to work on Domestic IOTs. Starting by defining a Webserver installed on a micro-controller which can also take inputs from the Raspberry Pi and other mobile devices. Further we configure to operate them on a single Wireless network. Using available tools like Dataplicity [Dat2019] we can command-line remote-access from anywhere around the globe.

Servers are costly to build and so is the maintenance cost of full-fledged workstations. The solution provided in the report will address the cost and convenience. It can be developed in almost 1/4th the actual amount of finance and resources required.

IOTs have their own challenges, like incompatibility issues, lack of standardization and security. Here, in this report we try to address the most important issue that these "things" are facing is the security. We try to make the connection between the client-server secure by installing firewall and further configuring it with set of rules. Also we make sure the "things" are configured properly by changing their default manufacturer ID's and login credentials. SSH and other physical port not left open. In this project we assume that the hardware used remains inaccessible to naive/unintended users and that the property security steps are already taken.

For a bigger environment (*Figure 3.5*) Raspberry Pi Cluster/stacks can be used and multiple sensors can be attached to the micro-controller. It is important to note that with multiple devices gaining access to the same wireless network there can be an arising issue of network congestion. Which should be address by changing the bands from the conventional 2.5 GHz to 5 GHz. If that is not feasible, access-points are required.

By adding relays and proper connectivity the proposed system in the report has almost all the attributes necessary for further designing and developing IOT smart home system [Yin2013], with multiple things attached and communicating with each other. Implementation and challenges are further explained in the later part of the report where it would be covering the specific use of software and hardware also the programming languages used, its compilation and outcomes. Below tables (*Table 1.1*) & (*Table 1.2*) shows the operating modes in brief for the micro-controller & boot-sequences for the micro-processor:

Micro-controller [ESP8266]	
Modes	**Functions**
Station Mode	WiFi.begin(ssid, password)
	WiFi.begin()
	WiFi.begin(ssid, password, channel, bssid, connect)
	WiFi.config(local_ip, gateway, subnet, dns1, dns2)
Soft Mode	WiFi.softAP(ssid)
	WiFi.softAP(ssid, password, channel, hidden)
	softAPConfig (local_ip, gateway, subnet)

Table 1.1: Micro-controller Modes & Functions [Onl2018]

Raspberry pi [3]	
Boot Type	**Description**
SD Card Boot	The default way of using a Raspberry Pi is to boot it using an SD card: this is the recommended method for new and inexperienced users.
USB Boot	Boot as a USB host using one of the following: *Mass Storage Boot* – boot from a mass storage device. *Network Boot* – boot via Ethernet.

Table 1.2: Micro-processor Boot Type & description [Ras2019]

2 LITERATURE REVIEW

It was year 1999 the term "Internet of Things" was coined by Kevin Ahston. Since then it has seen a major boom as these tiny devices continued to shrink in size while continuously offering increasing functionality. Today most of the Technology giants are in competition on investing in this future technology. Below figure/Graph (*Figure 2.1*) [Ava2018] depicts the emergence.

Internet of Things abbreviated as IOTs find implementations in almost all the technological fields know to humankind. They are used for both domestic as well as for industrial purpose and they continue to grow.

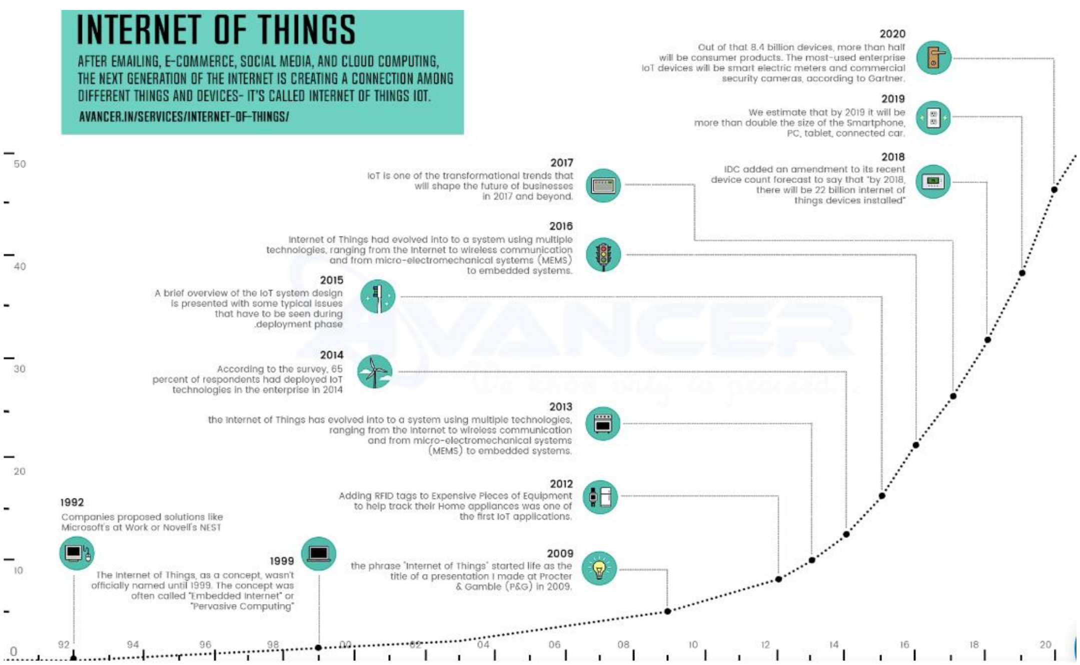

Figure 2.1: Emergence of Internet of Things On Timeline [Ava2018].

Since IOTs provide a cheap & flexible hardware, they can be good solution to testing and development purposes. The projects derives this quality & tries to introduce and further attempt to develop it. The Report emphasizes on providing these benefits to small industries & organizations inexpensive solutions for researching & developing their products and tools.

Each technology has its own benefits and downfalls. IOTs are no different. While numerous miniature “Things” can be configured. It becomes a challenge to make them communicate with each other evenly. This is because there in no particular standard established for doing so. Devices may use their own communication protocols and standards. Furthermore when connected they face a most concerning security issue. Making these numerous device secure under a single closed environment is not that difficult but, when connected to internet it’s a different story! Below table (*Table 2.1*) [Ana2019] shows the increasing number of devices connecting to internet since year 1990.

Year	**Number Of Connected Devices**
1990	0.3 million
1999	90.0 million
2010	5.0 billion
2013	9.0 billion
2025	1.0 trillion

Table 2.1: Boom In The Increasing Number Of Devices Connected To Internet [Ana2019].

To secure these devices sophisticated solutions must be reviewed which should not only be practical but cost-effective too. In our solution, we propose using internet of things [Nei2004] firewall for securing and administrating the inbound and outbound traffic. We intended to make use of open-source “uncomplicated firewall” abbreviated as UFW. The firewall does not impose much processing overhead on the device and is simple making it a perfect option for implementing it own low powered IOT micro-processors. It can be operated through both a GUI or through simple command line (shell). The Downfall of using the UFW is that it is incapable of providing some of the important firewall functionalities that full-fledged dedicated firewalls do like: Session based traffic inspection, block specific web traffic based on Ip. Other than that for a controlled & tightly bound environment it is a good option. For more serious application one should avoid using it.

3 METHODOLOGY

Introduction:

This section will cover the details, flow and implementation of methods utilized for building this project. To lay a foundational basics of the project, we start by deriving it from the Software Development Life-cycle and then proceed further step-by-step:

3.1.1 Requirement Analysis

3.1.2 Design

3.1.2 Implementation

3.1.2 Testing

3.1.2 Evolution/Maintenance

3.1 Requirement Analysis

Developing an IOT solution which would introduce how IOTs can be used as a cost-effective replacement of the native hardware for the purpose of testing, development and research. We assume the implementation small scale infrastructures or SOHOs. The project intends to implement a small portable server which would not only be able to host a small website but also infer IOT communications acting both as a server and a micro-controller. Further, assuming a limited budget, affording a dedicated workstation for communicating with a server is not an option. Last but not the least, efforts should be made to make the communication and the system secure.

3.2 Design

We first require a micro-controller for our project which would act both: server & micro-controller. We establish a micro-processor dedicated for communicating with it. Both server-Client communicate over a shared wireless LAN.

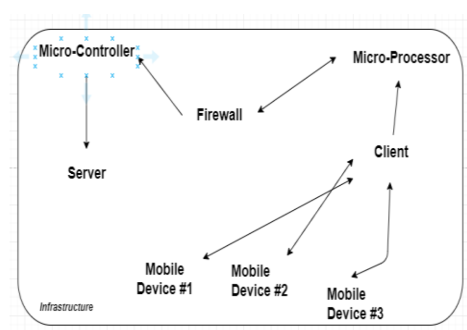

Figure 3.1: Design Overview

Thus, no physical interfacing required. In order to provide security [Udd2016], we configure the shared wireless LAN with SSID and password. Install Firewall on the workstation/server. A design overview is visible in figure (*Figure 3.1*).

3.3 Implementation

Below figure/screenshots (*Figure 3.2*) shows the actual selection of devices for the implementation. Where it is important to note that the sensors may remain variable as they can be either a LED or an actual electronic bulb connected to a relay. Further hardware screenshot/figure (*Figure 4.5*) is elaborated under the Outcome section **4**. We also enlist the possible and most preferred options that can be used to build the project. They are enlisted in the below tables:

Table 3.1: Most Popular Hardware Options
Table 3.2: Most Preferred Software Options
Table 3.3: Most Preferred Programming Language Options
Table 3.4: Other Options

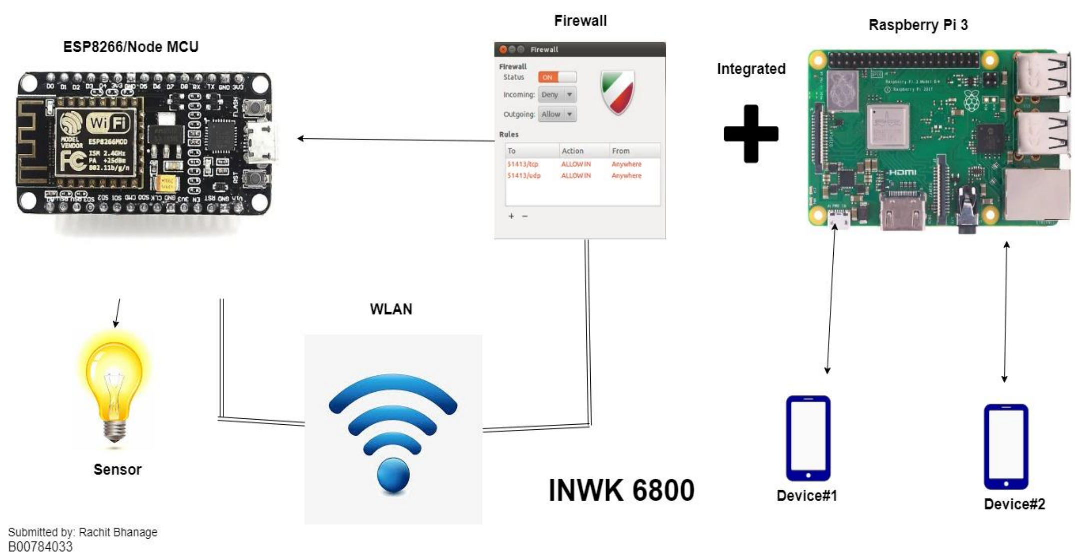

Figure 3.2: Device Selection

3.3.1 Hardware/Software

We have certain available hardware and Software listed below (Table 3.1), (Table 3.2), (Table 3.3). Out of which we select the appropriate hardware, reasoning provided below:

Section: Hardware Options		
Device Type	**Device**	**Specifications**
Micro-processors:	Beagle Board	512MB DDR3 RAM 4GB 8-bit eMMC on-board flash storage 32-bit instruction
	Raspberry Pi	1GB DDR3 RAM 4GB 8-bit eMMC on-board flash storage 64-bit instruction
Micro-controllers:	Arduino Uno	Operating Voltage : 5V Flash Memory :32 KB (ATmega328P) of which 0.5 KB used by bootloader Clock Speed: 16 MHz Length 68.6 mm Width 53.4 mm Weight 25 g
	ESP8266	Operating Voltage: 3.3V Flash Memory: 4 MB Clock Speed: 80 MHz Length 49 mm Width 24.5 mm Weight 12 g
	Arduino Mega 2560 Rev3	Operating Voltage : 5V Flash Memory 32 KB (ATmega328P) of which 0.5 KB used by bootloader Clock Speed: 16 MHz Length 68.6 mm Width 53.4 mm Weight 25 g

Table 3.1: Most Popular Hardware Options

Section: Base Operating Systems		
System	**Type**	**Version**
Operating System	Raspbian	Jessie Stretch Noobs Lite
Firewall	UFW	N/A
	IPCop	2.1.9

Table 3.2: Most Preferred Software Options

Section: Programming Language		
Purpose	**Name**	**Version**
Front-end use	HTML	5
	JavaScript	ECMAScript 2017 ECMAScript 2018
Back-end use	C	C99 C11
	C++	C++14 C++17
	Python	2.5 3.0

Table 3.3: Most Preferred Programming Language Options

Section: Other		
Type	**Name**	**Version**
Virtual Machine	VMware	VSphere ESXi 6.7. VSphere ESXi 6.5. VSphere ESXi 6.0.
Software Development Kit (SDK)	Arduino	1.8.9

Table 3.4: Other Options

3.3.2 Elaboration:

Step: 1

ESP8266 NodeMcu is configured by programming it with Arduino SDK (v1.8.9).
It is written in C++. We start by initially providing the NodeMcu connectivity with the WLAN. Appropriate SSID & Password are provided. Then further HTML5 code is introduced. Once NodeMcu [Jin2005] is connected successfully it provides an output through its serial monitor.

Step: 2

Raspberry Pi3B connected to the same network.
It is loaded with Jessie Stretch with SSH and a static Ip address.
UFW Firewall is configured with appropriate inbound/outbound rules as required by user. Graphical User Interface is also available for the same.
Finally using a python code, the firewall functionalities and the communication with esp82666/NodeMcu server is configured all in a single piece of code with a command- line interface. Proper security loop is maintained as Login credentials and Password are required to access the Code for operating the system.

Step: 3

A Shared network is created a mobile hotspot is also feasible for the purpose. Mobile Devices connected to the same network can access the webserver and sensors through the configured Raspberry Pi firewall. Indefinitely any compatible sensor can be connected to ESP8266/NodeMcu [Jin2005] if programmed. In this Project it's a LED.

3.1.2 Testing

The ESP8266 Connected to WLAN.

Its SSID and Password configuration as displayed in the below diagram (*Figure 3.3*).

The connectivity is derived from the ESP8266WIFI library.

Other respective libraries for time, Client/Server operations are defined.

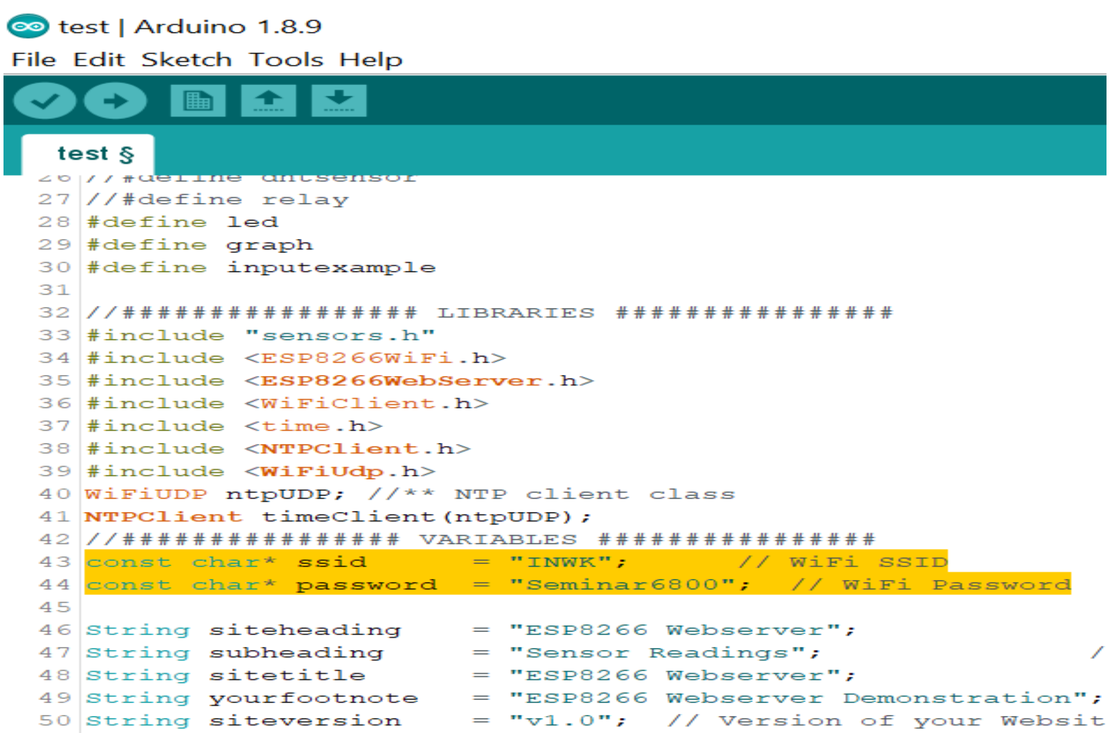

Figure 3.3: Arduino SDK/C++ Wireless Connection Code

- Firewall runs perfectly and does it jobs.
- Below is the figure (*Figure 3.4*) that show reflects the running status and rules configured.
- It requires the super-user permission to access.
  As mentioned SSH is enabled.

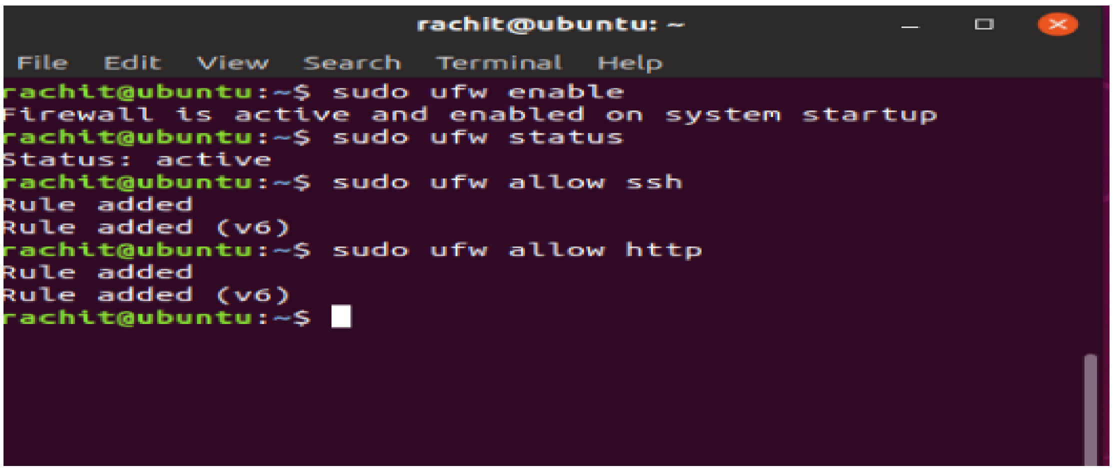

Figure 3.4: UFW Firewall Configuration

3.1.2 Evolution/Maintenance

The system has a huge scope of further development.

Below is the list of functionalities/sensors that can be further attached to micro-controller:

- Servo-motor Positioning & Display
- Controlling Relays
- Processing user inputs
- Display data using google charts
- DHT11 & DHT22 Temperature sensors
- Controlling multiple LEDs

The connection between Raspberry Pi and ESP8266 can be made more secure by refining the firewall rules further. A graphical user interface can be developed for the command-line program. Global connectivity can be provided by introducing port-forwarding.

Below figure (*Figure 3.5*) shows the possible utilization of the configuration to set-up full-fledged laboratory/environment.

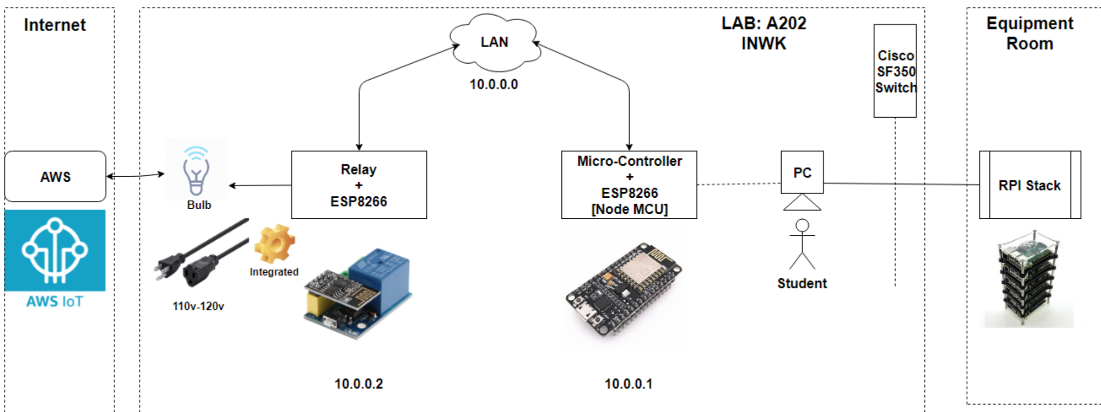

Figure 3.5: Laboratory Setup

4 CONCLUSION & RECOMMENDATIONS

After selecting specific library & selecting Generic NodeMcu [Jin2005] option, we are able to compile the code. The syntax of the C++ remains uniform throughout. As evident from the figure/screenshot (*Figure4.1*) below we are also making use of HTML in order to display the Website.

Figure 4.1: Successful compilation of the HTML5 & C++ code under the Arduino SDK environment

The Figure/Screenshot (Figure4.2) below shows the initial phase of the python program that the users needs to go through in-order to access the ESP8266 server. It is compiled with Python Version 3

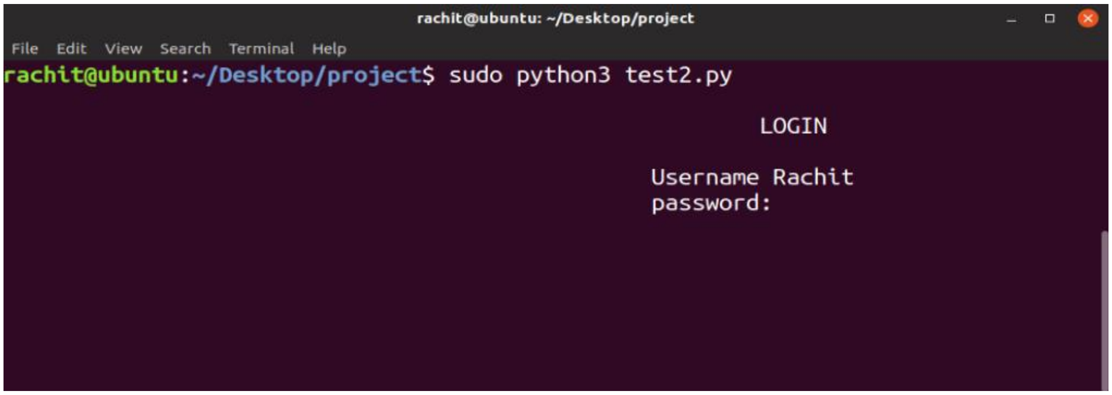

Figure 4.2:Front-End Command-line access mechanism for controlling the ESP8266 from the server (The python code is running on the server)

Below Figure/Screenshot (*Figure4.3*) shows the auto-configure option that is programmed in python to help user connect to server in event of connection-failure. It simple does by executing Linux system commands. Internet connectivity is required.

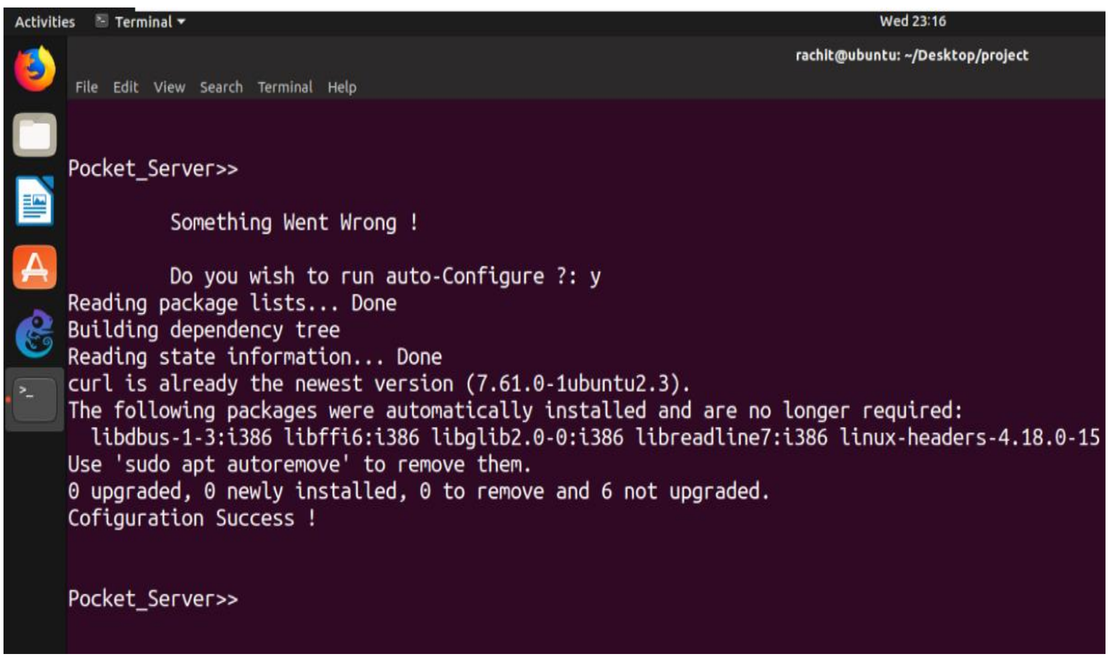

Figure 4.3: Implementation of Auto-Configuration Option In The Event of Failure Connecting with The Micro-Controller/Server

Uncomplicated Firewall rules can be set through the GUI provided or manually through command line. Below Figure/Screenshot (*Figure4.3*) shows the list of rules configured on the Raspberry Pi server. For the simplicity sake, of development, we are by default allowing all the traffic which later would be blocked according to the user needs or requirements. The benefit of using UFW is that it also supports ipv6 [Scr2019] as evident from below:

```
Status: active
Logging: on (medium)
Default: deny (incoming), allow (outgoing), deny (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW IN    Anywhere
80/tcp                     ALLOW IN    Anywhere
443/tcp                    ALLOW IN    Anywhere
25/tcp                     ALLOW IN    Anywhere
587/tcp                    ALLOW IN    Anywhere
143/tcp                    ALLOW IN    Anywhere
993/tcp                    ALLOW IN    Anywhere
53                         ALLOW IN    Anywhere
22/tcp (v6)                ALLOW IN    Anywhere (v6)
80/tcp (v6)                ALLOW IN    Anywhere (v6)
443/tcp (v6)               ALLOW IN    Anywhere (v6)
25/tcp (v6)                ALLOW IN    Anywhere (v6)
587/tcp (v6)               ALLOW IN    Anywhere (v6)
143/tcp (v6)               ALLOW IN    Anywhere (v6)
993/tcp (v6)               ALLOW IN    Anywhere (v6)
53 (v6)                    ALLOW IN    Anywhere (v6)
```

Figure 4.4: UFW Firewall Rules (By default set open for initial communication)

The Final physical/hardware view of the project is displayed in below figure/screenshot (Figure4.3). It contains following:

- NodeMcu
- Raspberry Pi3B
- Power Supply 3.8 volts
- Micro USB cables x 2
- HDMI connector
- Wireless USB Device
- Camera Module (Optional)

Figure 4.5: Final hardware/physical implementation of the project

The figures/screenshots below shows the actual design of the HTML5 webpage running on the ESP8266 micro-controller. Figure (*Figure 4.6*) displays the LED on/off status displayed when triggered through mobile device or Raspberry Pi Wireless.

Figure 4.6: LED Trigger Status Display

The Webpage on the webserver is also designed to take user inputs if sensors or actuators are used. It is capable of displaying he user input values as shown in the below two figures/screenshots (Figure 4.7) and (Figure 4.8):

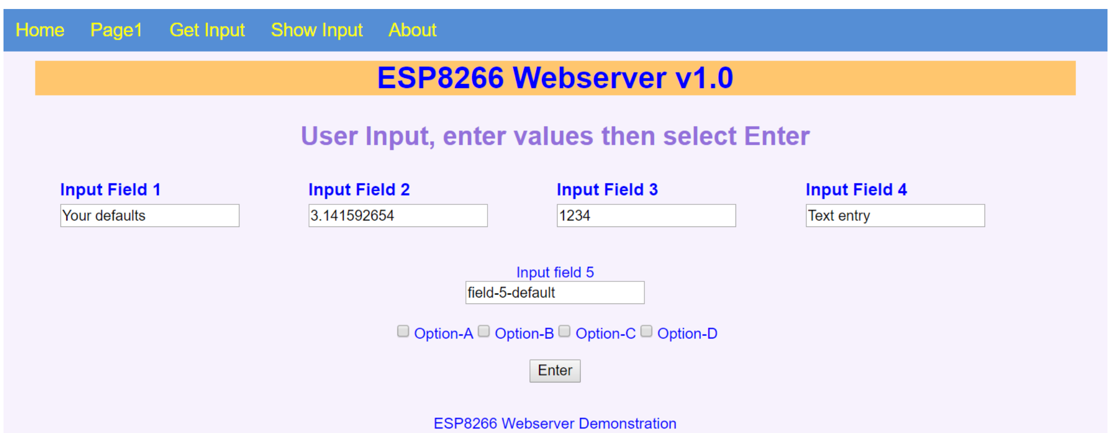

Figure 4.7: Taking User Input

Figure 4.8: Displaying User Input

REFERENCES

[Ava2018] "The History of IoT (Internet of Things) and How It's Changed Today," *Avancer Software Solutions*, 12-Feb-2018. [Online]. Available: http://avancer.in/history-iot-internet-things-changed-today/. [Accessed: 20-Apr-2019].

[Ana2019] Kashyap, "10 Real World Applications of Internet of Things (IoT) - Explained in Videos," *Analytics Vidhya*, 11-Mar-2019. [Online]. Available:https://www.analyticsvidhya.com/blog/2016/08/10-youtube-videos-explaining-the-real-world-applications-of-internet-of-things-iot/. [Accessed: 20-Apr-2019].

[Sta2019] "Global IT hardware spending forecast 2013-2019 | Statistic," *Statista*. [Online].Available:https://www.statista.com/statistics/422802/hardware-spending-forecast-worldwide/. [Accessed: 20-Apr-2019].

[Yin2013] Jie, Yin, et al. "Smart home system based on iot technologies." *2013 International Conference on Computational and Information Sciences*. IEEE, 2013.

[Raf2004] Gershenfeld, Neil, Raffi Krikorian, and Danny Cohen. "The internet of things." *Scientific American* 291.4 (2004): 76-81.

[Sta2019] "Cisco research and development expenses 2011-2018 | Statistic," *Statista*.[Online].Available:https://www.statista.com/statistics/350548/cisco-research-and-development-expenses/. [Accessed: 20-Apr-2019].

[Jin2005] Wang, Jingcheng, et al. "CAN/LIN hybrid network for automob ile." *IEEE International Conference on Vehicular Electronics and Safety, 2005.*. IEEE, 2005.

[Udd2016] “Security Wireless Sensor Networks By Mohd Faraz Uddin,” *Bartleby*, 28-Jan-2016.[Online].Available: https://www.bartleby.com/essay/Security-Wireless-Sensor-Networks-By-Mohd-Faraz-FKUPPJHW3FV8X. [Accessed: 20-Apr-2019].

[Scr2019] “ipv,” *Scribd*.[Online].Available: https://www.scribd.com/document/341551020/ipv. [Accessed: 20-Apr-2019].

[Bal2005] Subrahmanya, MH Bala. "Small-scale industries in India in the globalisation era: performance and prospects." *International Journal of Management and Enterprise Development* 2.1 (2005): 122-139.

[Onl2018] “How to use ESP8266 Wifi Modes: Station and Access Point,” *Online Shouter*, 13-Feb-2018. [Online]. Available: http://onlineshouter.com/use-esp8266-wifi-modes-station-access-point/. [Accessed: 20-Apr-2019].

[Ras2019] “Raspberry Pi boot modes,” *Raspberry Pi boot modes - Raspberry Pi Documentation*, 2019.[Online]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/. [Accessed: 20-Apr-2019].

[Dat2019] “Control your Raspberry Pi from anywhere.,” *Dataplicity*. [Online]. Available: https://www.dataplicity.com/. [Accessed: 20-Apr-2019].

BIBLIOGRAPHY

Majumder, S., Aghayi, E., Noferesti, M., Memarzadeh-Tehran, H., Mondal, T., Pang, Z., & Deen, M. (2017). Smart homes for elderly healthcare—Recent advances and research challenges. *Sensors*, *17*(11), 2496.

Ismail, W., & Hassan, M. (2017). Mining productive-associated periodic-frequent patterns in body sensor data for smart home care. *Sensors*, *17*(5), 952.

Rashidi, P., & Cook, D. J. (2013). COM: A method for mining and monitoring human activity patterns in home-based health monitoring systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *4*(4), 64.