

Wireless Smart Switch For Lighting Control Using NodeMCU and MQTT Protocol

Summer Internship

at C-DAC, Pune



PERIOD OF INTERNSHIP

11th MAY 2016 - 11th JULY 2016

Project Guide: Mr Tarun Deep Kohli

Signature:

Submitted by

Archana Kumari,

Roll No.1301013

B.Tech.

E&CE Dept., IIIT Guwahati

8th JULY 2016

Preface

This report documents the work done during the summer internship at Centre for Development of Advanced Computing (C-DAC), Pune under the supervision of Mr Ashish Kuvelkar, Tarun Deep Kohli and Mayur Kulkarni. The report first shall give an overview of the tasks completed during the period of internship with technical details. Then the results obtained shall be discussed and analyzed.

Report shall also elaborate on the future works which can be persuaded as an advancement of the current work.

I have tried my best to keep report simple yet technically correct. I hope I succeed in my attempt.

Archana Kumari

Acknowledgement

Simply put, I could not have done this work without the lots of help I received cheerfully from whole C-DAC. The work culture in C-DAC really motivates. People are very helpful here. Everybody is such a friendly and cheerful companion here that work stress never comes in way. I am also highly indebted to my supervisor Mr Tarun Deep Kohli and Mr Mayur Kulkarni, who seemed to have solutions to all my problems. I would specially like to thank Mr Ashish Kuvelkar for proving the nice ideas to work upon. Not only they advised about my project but also promoted me to discuss on it and gave adequate time to familiarise myself to it. Meetings and discussions with them have evoked a good interest in the project idea. I am also thankful to Mr Azeezur Rahman, for working with a team spirit with me in the whole project.

I also express my gratitude to Mr Prasad Wadlakondwar, Mr Abhishek Das, Mrs Shweta Das and others for their help and support at C-DAC. C-DAC being my first interface to professionalism, has helped me learn qualities and discipline of it.

Archana Kumari

PROJECT DOCUMENTATION

Abstract:

This project is intended towards a kind of home automation and internet of things (IoT). This documentation aims to demonstrate the insights of the works and implementations in this regard. It will deal with an introduction to what the requisite of the project are and what may be the problems with them and how to handle them.

Keywords:

NodeMCU, ESP8266, MQTT, Timer, Interrupts.

1. INTRODUCTION:

Objective of our work is to build a wireless switch which can work inside a network to control office/home lights or other devices. This switch is not intended for a single lights but as many as you want and also it can work for devices other than lights, but they need to be connected to the home network.

In this project, NodeMCU is used as an embedded platform and MQTT being a protocol defined for machine to machine (M2M) communication.

NodeMCU :

This project uses ESP8266^{[1][2][4]}, a low-cost Wi-Fi chip with full TCP/IP stack and microcontroller capability, integrated on a small embedded development board NodeMCU^{[3][5]}, which is an IoT platform. It is interactively programmable in Lua scripts. It can also be programmed in C using Arduino IDE^[8]. The documentation provide on nodeMCU clearly shows the pin configurations.

Features :

- 32-bit RISC MCU running at 80 MHz.
- 64 KB of instruction RAM, 96 KB of data RAM.

- External flash memory- 512 KB to 4 MB (up to 16MB is supported).
- Supports IEEE 802.11 b/g/n WiFi at 2.4 GHz
 - WEP, WPA/WPA2 authentication and Open Network feature.
- 16 GPIO pins.
- SPI, I²C.
- I²S interfaces with DMA (sharing pins with GPIO).
- UART on dedicated pins, plus a transmit-only UART can be enabled on GPIO2.
- One 10-bit ADC.

Now working with the nodeMCU in Arduino IDE looked similar as that when working with an arduino board in this IDE, but there were some exceptions too. Some of the functions and methodology that easily worked with arduino boards, were not at all supported with nodeMCU. Some extra libraries were also required to be added. Following are some of those odd things with this board and IDE which were among the exceptions:

- **PIN MAPPING** : When working with Arduino IDE, it is important to consider proper mapping of the pin numbers during coding programs. For example if it is pin D5 connected to GPIO14, then we have to put pin number 14 and not 5 in the program code. But an easy way to avoid such mapping confusions, is to use `#define ledPin D5`, which also works fine.
- **WiFi** : Next step into it was to learn about WiFi library for WiFi access of Esp8266. `WiFi.begin(ssid, password)`, `WiFi.status()`, `WiFi.disconnect()`, `WiFi.localIP()`, and etc. were used in this aspect.
- **POLLING Vs INTERRUPT** : As the project was mostly centered around obtaining different functionality of pushbuttons on the board, it was important to check button status and state if it was a short press, long press, still pressed or released. Initially this reading of button state was done on pooling basis, that is, every time in the main loop function, the program checks the state of the buttons. Since this polling is never

considered an efficient way to do this, the next challenge was to find interrupt method for this. It was found that the interrupt function used for arduino boards worked well with this board too. Functions like `attachInterrupt(pin, function, mode)`, `interrupts()`, `detachInterrupt()`, `noInterrupts()` which work for an arduino boards, were found to work well with this board also. So making interrupt calls became easy.

- **BUTTON PRESS** : Now the next thing to be done was to identify and differentiate between a short press or a long press. In this step, the challenge was to implement a timer interrupt which could trigger up just after a second of pressing the button and call an ISR (Interrupt Service Routine) to check if any falling edge of the button was encountered. If there was a low state of button before the timer interrupt came, it was considered as a short press or else it was a long press at hold situation until the interrupt for the falling edge of the button was found.
- **TIMER INTERRUPT** : But it was not easy to find a timer function which could work for nodeMCU in Arduino IDE. Timer0 was found to work fine with it. It could work well as a timer interrupt and hence the problem was resolved.
- **JITTER** : Now since button state was checked on an interrupt basis, one problem which could come was to handle jitter. So one easy trick to handle it was to ignore the state changes within 20 ms or some small time.

MQTT :

After getting familiar to the board, a protocol which would make machine to machine communication possible, was learnt. MQTT^{[6][7]} was firstly proposed by Andy Stanford-Clark (IBM) and Arlen Nipper (Arcom, now Cirrus Link) back in 1999.

MQTT, abbreviated as Message Queuing Telemetry Transport, is such protocol which uses publish and subscribe mechanism. It uses binary format (with shorter headers), requiring less bandwidth and battery. It is a Client Server publish/subscribe messaging transport protocol. In

this protocol, there are clients connected to an MQTT server or broker, such that all clients are directly connected to the broker but no two clients are connected together and they have no idea of each other. These clients can either subscribe or publish or both on a topic or more topics. The message published by a client on a topic is sent to all the other clients who have subscribed for that topic and are connected to the broker. And this controlled distribution of messages based on their topic interests, is the sole responsibility of the MQTT broker and no other client has to do anything else than publish and subscribe. Hence, this protocol is very much useful of machines(clients) having restricted battery usage issue. In this protocol, a good point is that the protocol do not ensure the best service every time, the quality of service (QoS) is defined by user, and hence it does not waste its power in trying to achieve best service possible, which makes it simple and reliable. Hence,

➤ **Features of MQTT:**

- MQTT is a Client Server Publish/Subscribe messaging transport protocol.
- Simple to implement.
- Provides a Quality of Service Data Delivery.
- Lightweight and Bandwidth Efficient.
- Data Agnostic.
- Continuous Session Awareness.

- **The publish/subscribe pattern**

The publish/subscribe pattern (pub/sub) is an alternative to the traditional client-server model, where a client communicates directly with an endpoint. However, Pub/Sub decouples a client, who is sending a particular message (called publisher) from another client (or more clients), who is receiving the message (called subscriber). This means that the publisher and subscriber don't know about the existence of one another. There is a third component, called broker, which is known by both the publisher and subscriber, which filters all incoming messages and distributes them accordingly.

The clients are nothing but the devices working as Input/ Output or both. Like temperature sensor (Input device to read temperature), LCD display (Output device to display the temperature) or a Smartphone (used to send messages as well as receive readings from temperature sensor).

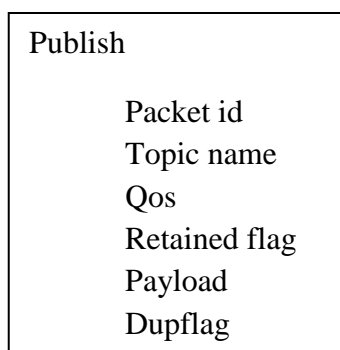
Main aspect in pub/sub is the decoupling of publisher and receiver, which can be differentiated in more dimensions:

1. **Space decoupling:** Publisher and subscriber do not need to know each other (by IP address and port for example)
2. **Time decoupling:** Publisher and subscriber do not need to run at the same time.
3. **Synchronization decoupling:** Operations on both components are not halted during publish or receiving.

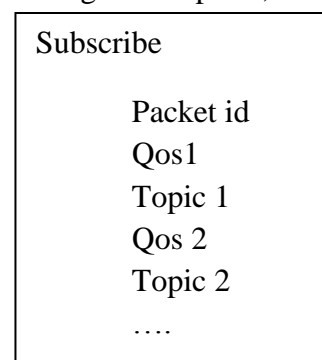
In summary publish/subscribe decouples publisher and receiver of a message, through filtering of the messages it is possible that only certain clients receive certain messages. The decoupling has three dimensions: Space, Time, Synchronization.

Firstly, to proceed with this protocol, an MQTT broker(Hivemq) was setup in the laptops. There are also other brokers available, some of them being from Mosquitto, ActiveMQ, Emqttd, etc. And for client setup, only a library for mqtt client was downloaded and included in Arduino IDE. Also an Eclipse paho client or MQTTLens client can be setup in laptops. After setting up these above requisites for the protocol, some keywords {connect, disconnect, subscribe, unsubscribe, publish, etc, }and functions were learnt.

For a publishing packet, the contents will be



For a packet seeking subscription, contents



Message Filtering :

It is important for the broker to filter out the messages carefully such that it does not messages to a client for which it is of no interest.

- **Option 1: Subject-based filtering**

The filtering is based on a subject or topic, which is part of each message. The receiving client subscribes on the topics it is interested in with the broker and from there on it gets all message based on the subscribed topics. Topics are in general strings with an hierarchical structure, that allow filtering based on a limited number of expression.

- **Option 2: Content-based filtering**

Content-based filtering is as the name already implies, when the broker filters the message based on a specific content filter-language. Therefore clients subscribe to filter queries of messages they are interested in. A big downside to this is, that the content of the message must be known beforehand and cannot be encrypted or changed easily.

- **Option 3: Type-based filtering**

When using object-oriented languages it is a common practice to filter based on the type/class of the message (event). In this case a subscriber could listen to all messages, which are from type Exception or any subtype of it.

Also there can be situations when a client connected to broker which has subscribed for some topics, is currently not connected to the broker due to some improper connection loss. Then there can be two ways to handle such situation:

1. The broker saves only the last value/message of each topic in its buffer and flow these messages to that client once the connection is re-established.
2. Other way is to save all the messages in the broker buffer while the client was offline.

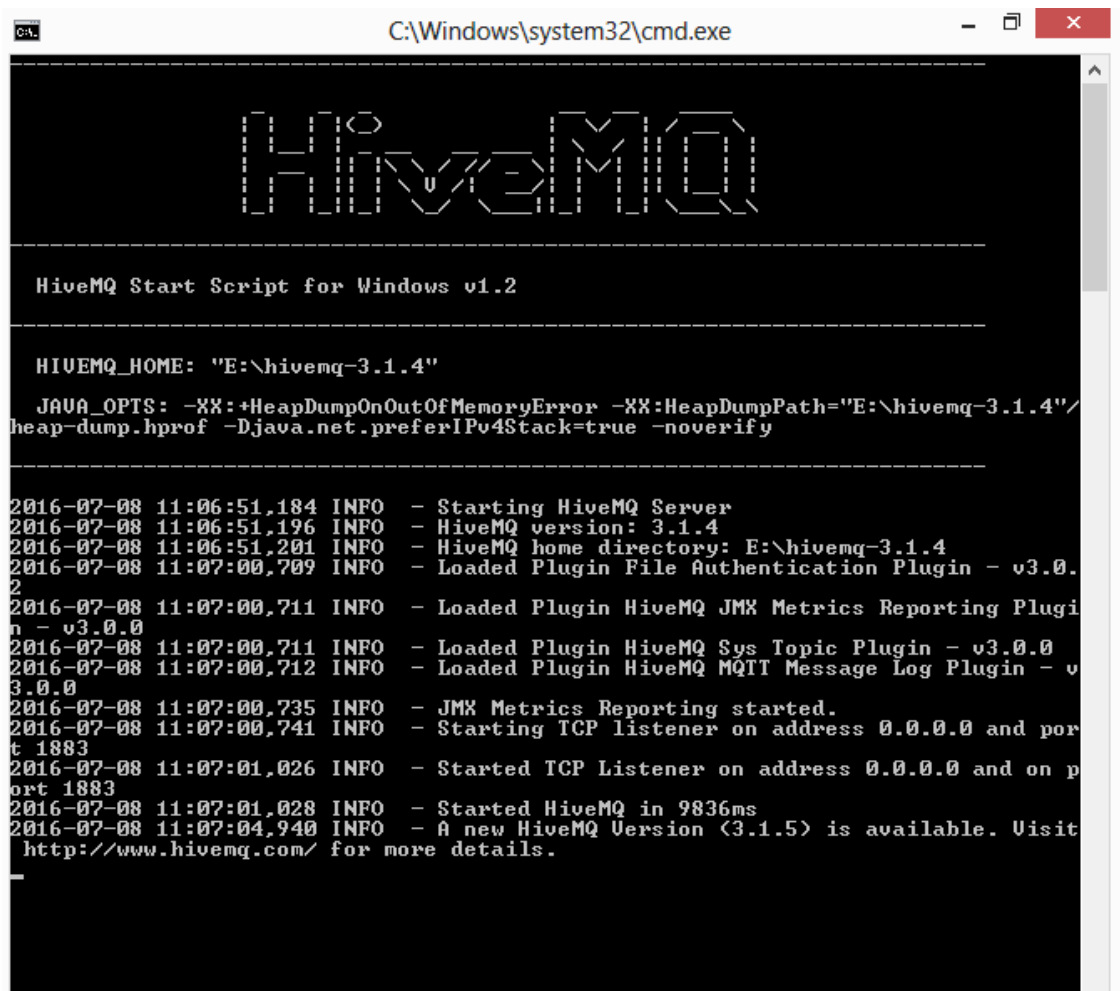
2. WORKING METHODOLOGY:

After learning different aspects of this project, the real experience was obtained in programming for its implementation. Following steps were kept in mind during programming:

1. Essential libraries were added or included in the program and all the required global variables were defined.
2. A setup function was written for setting up the serial communication, pin modes, connection to wifi and connection to MQTT server/ broker.
3. Also interrupts to the button pins were attached and corresponding ISRs were defined.
4. It was taken care that no network related functions were held in the ISRs, which may cause a delay in them.
5. All such publish, subscribe and callbacks functions were not related to any ISRs.
6. For determining whether a button press was short or long, timer interrupt was used. With every rising edge interrupt of the button, the timer interrupt of 1second was started and after which button state was read. If it was high, a long press which is still at hold was considered. Otherwise, it was declared a short press.
7. For every short press, only one message need to be published.
8. While for a long press, messages for "hold" were published periodically as long as button was not released. Once the button was is released, one message signaling "long press released" was published.
9. Meanwhile, situations of WiFi disconnect or MQTT client server disconnect appeared which were handled by a short function to re-establish the connections.
10. All these publish were done in the loop function.

3. Results and Discussions:

After writing the sketch, it was verified (or compiled) in the IDE and the uploaded into the kit. Now meanwhile, the MQTT broker (Hivemq) was started in the computer.



```
C:\Windows\system32\cmd.exe

HiveMQ

-----
HiveMQ Start Script for Windows v1.2
-----

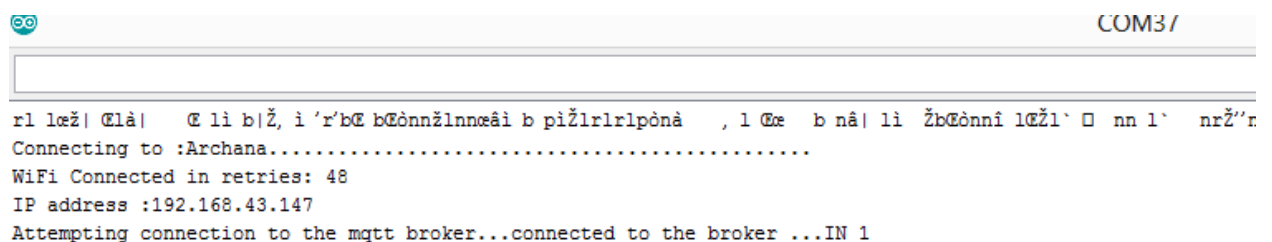
HIVEMQ_HOME: "E:\hivemq-3.1.4"

JAVA_OPTS: -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath="E:\hivemq-3.1.4\heap-dump.hprof" -Djava.net.preferIPv4Stack=true -noverify

-----
2016-07-08 11:06:51.184 INFO - Starting HiveMQ Server
2016-07-08 11:06:51.196 INFO - HiveMQ version: 3.1.4
2016-07-08 11:06:51.201 INFO - HiveMQ home directory: E:\hivemq-3.1.4
2016-07-08 11:07:00.709 INFO - Loaded Plugin File Authentication Plugin - v3.0.2
2016-07-08 11:07:00.711 INFO - Loaded Plugin HiveMQ JMX Metrics Reporting Plugin - v3.0.0
2016-07-08 11:07:00.711 INFO - Loaded Plugin HiveMQ Sys Topic Plugin - v3.0.0
2016-07-08 11:07:00.712 INFO - Loaded Plugin HiveMQ MQTT Message Log Plugin - v3.0.0
2016-07-08 11:07:00.735 INFO - JMX Metrics Reporting started.
2016-07-08 11:07:00.741 INFO - Starting TCP listener on address 0.0.0.0 and port 1883
2016-07-08 11:07:01.026 INFO - Started TCP Listener on address 0.0.0.0 and on port 1883
2016-07-08 11:07:01.028 INFO - Started HiveMQ in 9836ms
2016-07-08 11:07:04.940 INFO - A new HiveMQ Version (3.1.5) is available. Visit http://www.hivemq.com/ for more details.
```

Figure 1:Screenshot showing the starting of the MQTT server of Hivemq in the computer

Now the board connects to the WiFi.



```
COM3/

r1 lœž| Elà|  @ li b|ž, i 'r'b@ bEönnžlnœâi b pižlrlrlpònà , l @œ b nâ| li žbEönni lEžl' □ nn l' nrž'r
Connecting to :Archana.....
WiFi Connected in retries: 48
IP address :192.168.43.147
Attempting connection to the mqtt broker...connected to the broker ...IN 1
```

Figure 2: Screenshot showing the serial monitor of the Arduino IDE showing the connection setup of the kit

Now after connecting to the WiFi, it also connects to the MQTT server.

```
2016-07-08 11:07:01,026 INFO - Started TCP Listener on address 0.0.0.0 and on p
ort 1883
2016-07-08 11:07:01,028 INFO - Started HiveMQ in 9836ms
2016-07-08 11:07:04,940 INFO - A new HiveMQ Version (3.1.5) is available. Visit
http://www.hivemq.com/ for more details.
2016-07-08 11:16:13,511 INFO - No valid license file found. Using evaluation li
cense, restricted to 25 connections.
2016-07-08 11:16:13,598 INFO - Client ESP8266Client connected
2016-07-08 11:16:13,760 INFO - Subscribe from client ESP8266Client received: ON
QoS: 0
2016-07-08 11:16:13,848 INFO - Subscribe from client ESP8266Client received: OF
F QoS: 0
2016-07-08 11:16:22,972 INFO - Client ESP8266Client connected
2016-07-08 11:16:22,993 INFO - Client ESP8266Client disconnected
2016-07-08 11:16:23,114 INFO - Subscribe from client ESP8266Client received: ON
QoS: 0
2016-07-08 11:16:23,134 INFO - Subscribe from client ESP8266Client received: OF
F QoS: 0
```

Figure 3: MQTT server console showing connection to the kit with clientID "ESP8266Client" and subscription to the topics

Now the working of interrupt was tested.

a) Short Press of OFF button:

```
C:\Windows\system32\cmd.exe
2016-07-08 11:23:22,295 INFO - Loaded Plugin HiveMQ Sys Topic Plugin - v3.0.0
2016-07-08 11:23:22,296 INFO - Loaded Plugin HiveMQ MQTT Message Log Plugin - v
3.0.0
2016-07-08 11:23:22,332 INFO - JMX Metrics Reporting started.
2016-07-08 11:23:22,343 INFO - Starting TCP listener on address 0.0.0.0 and por
t 1883
2016-07-08 11:23:22,753 INFO - Started TCP Listener on address 0.0.0.0 and on p
ort 1883
2016-07-08 11:23:22,755 INFO - Started HiveMQ in 12707ms
2016-07-08 11:23:23,447 INFO - No valid license file found. Using evaluation li
cense, restricted to 25 connections.
2016-07-08 11:23:23,556 INFO - Client ESP8266Client connected
2016-07-08 11:23:23,763 INFO - Subscribe from client ESP8266Client received: ON
QoS: 0
2016-07-08 11:23:23,832 INFO - Subscribe from client ESP8266Client received: OF
F QoS: 0
2016-07-08 11:23:26,258 INFO - A new HiveMQ Version (3.1.5) is available. Visit
http://www.hivemq.com/ for more details.
2016-07-08 11:23:38,370 INFO - Client ESP8266Client sent a message to topic "OF
F": "100" (QoS: 0, retained: false)
```

Figure 4: Screenshot showing Message published on the topic"OFF" and the payload is 100 meaning a short press.

b) Long Press of ON Button

```
2016-07-08 11:25:01,533 INFO - Client ESP8266Client sent a message to topic "ON"
": "Dim Up" (QoS: 0, retained: false)
2016-07-08 11:25:01,434 INFO - Client ESP8266Client sent a message to topic "ON"
": "Dim Up" (QoS: 0, retained: false)
2016-07-08 11:25:01,533 INFO - Client ESP8266Client sent a message to topic "ON"
": "Dim Up" (QoS: 0, retained: false)
2016-07-08 11:25:01,636 INFO - Client ESP8266Client sent a message to topic "ON"
": "Dim Up" (QoS: 0, retained: false)
2016-07-08 11:25:01,731 INFO - Client ESP8266Client sent a message to topic "ON"
": "Dim Up" (QoS: 0, retained: false)
2016-07-08 11:25:01,832 INFO - Client ESP8266Client sent a message to topic "ON"
": "Dim Up" (QoS: 0, retained: false)
2016-07-08 11:25:01,933 INFO - Client ESP8266Client sent a message to topic "ON"
": "Dim Up" (QoS: 0, retained: false)
2016-07-08 11:25:02,032 INFO - Client ESP8266Client sent a message to topic "ON"
": "Dim Up" (QoS: 0, retained: false)
2016-07-08 11:25:02,130 INFO - Client ESP8266Client sent a message to topic "ON"
": "Dim Up" (QoS: 0, retained: false)
2016-07-08 11:25:02,231 INFO - Client ESP8266Client sent a message to topic "ON"
": "Dim Up" (QoS: 0, retained: false)
2016-07-08 11:25:02,310 INFO - Client ESP8266Client sent a message to topic "ON"
": "0" (QoS: 0, retained: false)
```

Figure 5: Screenshot showing messages published for a long press on the topic "ON" with payload "DimUp".

Now these results were of smooth functioning of the board. Following cases were also tested:

a) When WiFi connection disrupts in between:

The sketch is made in such a way that it checks the connection status all the time, and as soon as it detects a fault in connection, it attempts reconnection after finding the reason behind disconnection. If it fails to reconnect in specified number of attempts, it resets the kit and start from the initial state of set up.

```

Attempting connection to the mqtt broker.....
WiFi status : 3
MQTT client status : -2
Please check the MQTT Broker
Trying again after reset...

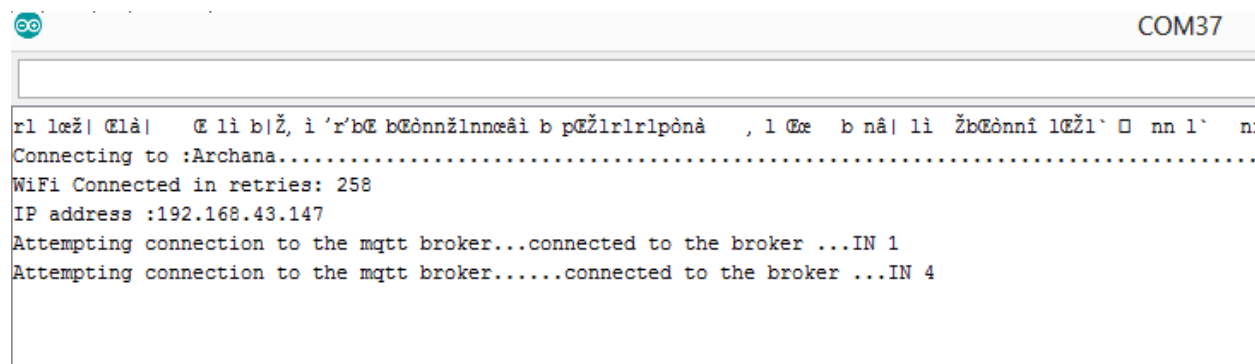
ets Jan  8 2013,rst cause:2, boot mode:(3,7)

load 0x4010f000, len 1384, room 16
tail 8
chksum 0x2d
csum 0x2d
v40106aac
~ld
Ø
Connecting to :Archana.....
WiFi Connected in retries: 370
IP address :192.168.43.147
Attempting connection to the mqtt broker....

```

Figure 6: Screenshot of Serial monitor showing attempts to connect the client after WiFi disconnect

b) If the MQTT server stops in between:



```

Connecting to :Archana.....
WiFi Connected in retries: 258
IP address :192.168.43.147
Attempting connection to the mqtt broker...connected to the broker ...IN 1
Attempting connection to the mqtt broker.....connected to the broker ...IN 4

```

Figure 7: Screenshot showing re attempt to connect after MQTT server stops in between

c) If both the buttons are pressed simultaneously:

When both buttons are simultaneously pressed, it checks which one was pressed first and handles interrupt of only that pin, and disables the other one as long as

the interrupt on the privileged pin does not end. Here for testing, first ON Button was pressed and kept on hold (long press still on hold), meanwhile OFF button was also long pressed along with that. But the program did not allow OFF Button interrupt, that is locked interrupt.

Figure 8: Screenshot of serial monitor showing locked interrupt state

The project was an exciting experience of building such device using noddeMCU. It took more time to search for solutions to the problems related to the kit and some of the APIs of it, because of less documentation available on it. But it proved to a good project for learning and implementation both.

In this project, the wireless switch is confined to work within an internal network. Working in an intra network is easy, but would require several permissions to work in an inter network situation. But since there was no requirement for an inter network switch, no work has been done on this aspect. One can see such objective as its future scope but ensuring proper security would be a challenge.

It is realised that this kit is very useful, has ability to work as an efficient microcontroller and a WiFi supporter and is of low cost too. Many different functionalities can be achieved with this kit and an efficient coding, if one has a nice idea/plan.

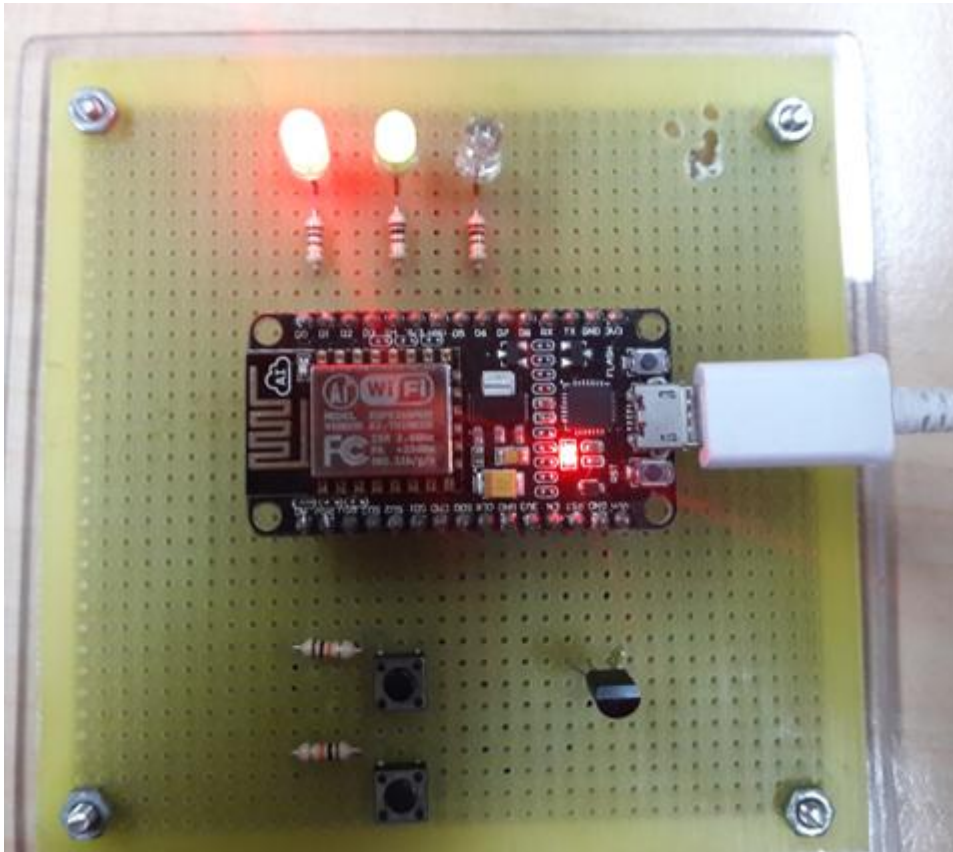


Figure 9: Figure showing the working nodeMCU Kit

5. References

- [1] (2016) *ESP8266EX Datasheet* -Version 4.3, Espressif Systems IOT Team, Espressif
- [2] Labradoc, *ESP8266 WiFi Module Quick Start Guide*, 2014,
<http://www.labradoc.com/i/follower/p/notes-esp8266>
- [3] NodeMCU Team, <http://nodemcu.com>
- [4] Espressif, *ESP8266 Technical Reference*, version 1.1 (2016)
- [5] NodeMCU Team(2015), *NODE MCU DEVKIT V1.0*, version 1.0, NodeMCU
- [6] Hivemq Enterprise MQTT Broker, <http://www.hivemq.com/>
- [7] PubSubClient, *API Documentation*,
<http://pubsubclient.knolleary.net/api.html#state>
- [8] ARDUINO, <https://www.arduino.cc/>