*Sai Lochan Reddy Reddipalli, Sasidhar Pasupuleti, Raghuram Kambhampati, Rohith Raj Reddy Alla Sai Poorna*

Mobile Cloud Supported Smart Light IoT System

[[1]](#footnote-1)

*Abstract*—This project helps develop a Mobile Application which enables cloud connection and control for a traditional Led Light. With this app the users can remotely configure the light by changing the schedule of operations, RGB patterns and the intensity. This is achieved through a user-friendly Mobile and Web Interface which also allows new user and device registration through a QR code for a secure connection. We used easily available hardware for accessibility and easy integration. Focus of this project is to make the user interface easy to use and seamless. Entire system is a surveillance-based Light with prime focus on control and maintenance. The interface works with multiple users controlling many devices at any given time. This makes the project scalable and efficient. The metrics we used in this project are performance, complexity, memory usability, security, stability and delay. We measured and tested the whole system for reliability and robustness making the product deployable for the industry.

***Index Terms*—IoT, Cloud, Smart Light**

# INTRODUCTION

T

HIS document describes the involved architecture, resource utilization and process implementation of our project. Our project deals with developing a mobile and web application to enable connection between Smart LED Lights and a cloud service. IoT as a concept plays a major role in our project. IoT as a connected service helps utilize the cloud space to run an instance and enable remove connectivity. In this project, we created a database to keep track of the session data, user data and device data. This helped us collect information and input from the user and push it through to the device. Both the user interface and the database read specific collection of data which sends values in a format understandable by the hardware to make the light act on the same and perform the required functions. Some guidelines we followed during this project are to use commodity software with freely available modules, built in memory to build the functions on the board and to use most economical hardware and software to implement the same. NodeMCU is the hardware we used to implement Wi-Fi connectivity for the device. It’s important for the light to connected to the internet at all times for the application to work and troubleshoot. Database schema is the next most important part of our project. We used MySQL DB to host our database. We connected this database with instance we ran on AWS EC2 server. This helps maintain all the records and also to send commands from our mobile devices to the database. This in turn enables the device to perform said functions. Our user interface is twofold, one being a web interface and the other a user interface. Scheduling hours of operation is one of the main functionalities. This helps keep track of the working of the light.

Other functionalities such as intensity and RGB pattern selection help with setting the exposure for the light. All these

functionalities are seamlessly integrated through the mobile and web interfaces. The user can register through these interfaces. Once registered, user can use the interface to add multiple devices and control them by using the settings specific to that device. These user commands are then tagged with user ids pertaining to the user so its user specific as well.

# Background

This section explains the motivation, problem statement, proposed system, and the requirements to develop a mobile cloud supported smart light IoT system.

## Motivation

IoT as a market is growing and traditional companies are trying to adapt to same to catch up to the speed. Economically speaking this project helps do the same for utilizing many of the IoT’s advantages. Market for Internet of things (IoT) is projected to hit $267 billion by 2020. Many traditional companies are trying to parlay IoT into connected services which can create tangible value and profit. Ever since mobile became more than a way of communication, mobile apps have added more value to users than any of us ever expected. IoT is one such technology that has immense potential to create constant value and absolute control for the users. IoT Mobile Applications create a refreshingly satisfying experience and through IoT enabled products. This brings in a huge customer database. Scalability, Data Management, Device Management and Security are some of the primary advantages of cloud-based applications. IoT is becoming a competitive differentiator amongst companies and hence the same needs to be implemented to bridge that gap. Also, the need for economical methods and hardware to implement is always necessary in the field of connected services. The more we try to reduce the cost of the system, the more accessible it becomes. So, will the profit margin for the same react in the same way. Traditional companies inculcating this technology is the main motivation behind this project.

## Abbreviations and Acronyms

LED – Light Emitting Diode

DB – Database

AWS – Amazon Web Services

IoT – Internet of Things

JSON – Java Script Object Notation

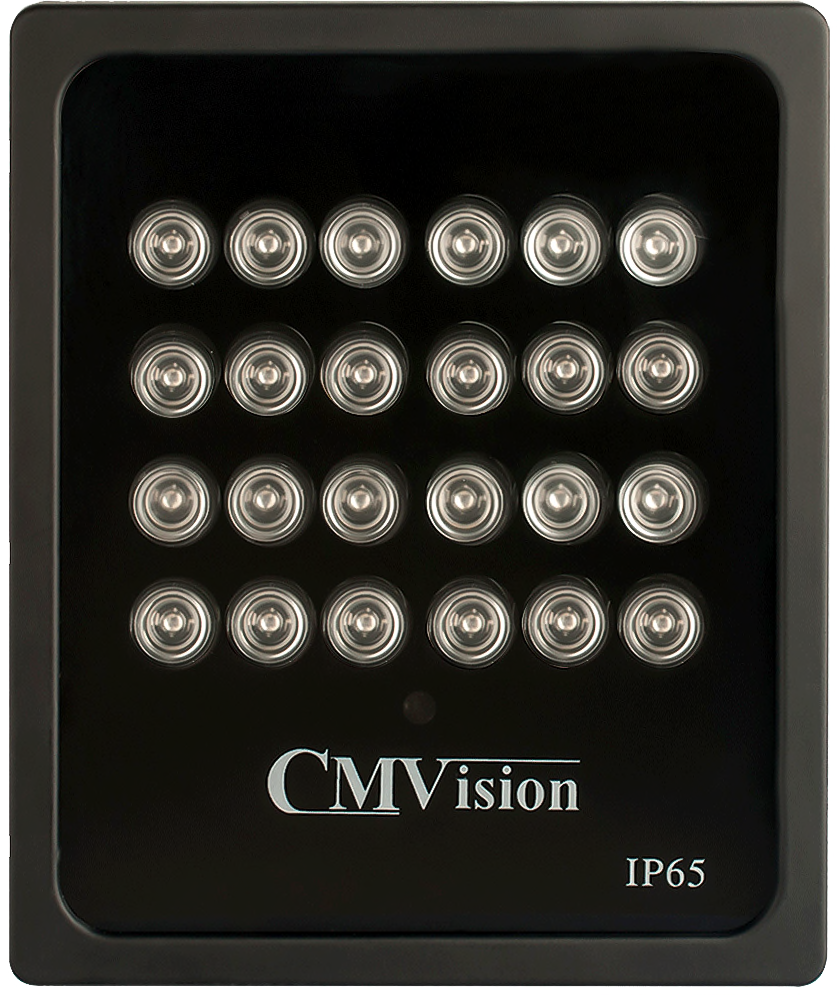
MCU – Micro-controller Unit

## Problem statement

There are many existing solutions regarding low light Infra-red sensor enabled lights that enhance night vision for cameras. There were many challenges being face din the market with the use of these technologies. First being exposure for the image, any object when far away from the camera was blurred and the one nearer was being over exposed to the light itself. Second, none of these solutions are currently integrated with the new age technologies such as IoT, a complete seamless interface. This when done opens up options to interact with the system and change settings on the go. Also, when an end-user platform is considered it can be tweaked to include new features and make a traditional light multifunctional in nature. The solution is to develop a LED light which is economical, has an optimal wavelength to solve exposure problems and which is cloud connected which allows remote access. All this also needs to be connected via a mobile application. This is necessary to tie all these features together. This application will help tweak the settings and control the light to get the right amount exposure. All this works out to be cheaper and more efficient than the existing solutions.

## Proposed System

* The system consists of CMVision Light, NodeMCU, AWS and Mobile application.
* The light is connected to NodeMCU and controlled by it.
* The mobile application will initially connect with NodeMCU configured as a hotspot.
* The user configures NodeMCU to connect with home Wi-Fi using the mobile application.
* The application holds a variety of options and settings to control the light. The user can select an option and can interact with the light through a mobile or web interface.
* The commands from the mobile and web interfaces go through the AWS cloud database to reach the light.





## Requirements

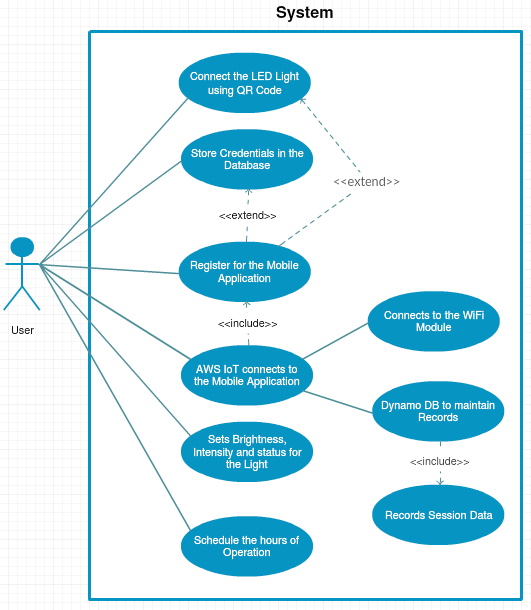
* Develop a Mobile Application with a seamless User Interface on the iOS Platform.
* Use Wi-Fi as the communication protocol between the devices.
* Connect the Light with NodeMCU.
* Provide a secure connection by registering users and adding devices through QR code.
* Enable remote access for controlling the light through the internet.
* Create a database to collect session data and enable cloud connection.
* Read light status, change RGB pattern, change intensity and set schedule for the light through a mobile and web interface.

# Design

Design Analysis for the project includes three modeling schematics to describe the functionality and layout of the entire product. This project includes multiple components such as AWS, iOS Mobile Application and the Wi-Fi Module connected LED Light. These different components work individually but function in complete synergy to let the user control the product with ease.

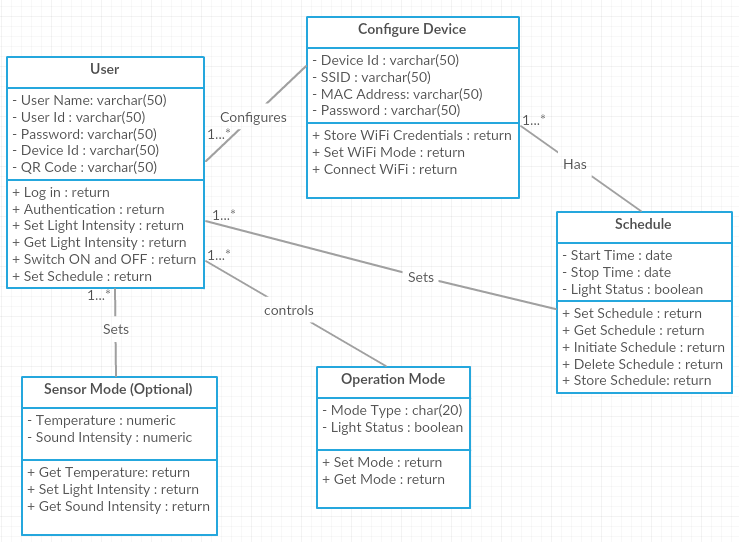
## Behavioral Modeling

User interaction with the system and how the system responds to the same is recorded in this. Use cases defining each action and what it includes or extends to is seen here. Each use case describes interaction, functionality and response.



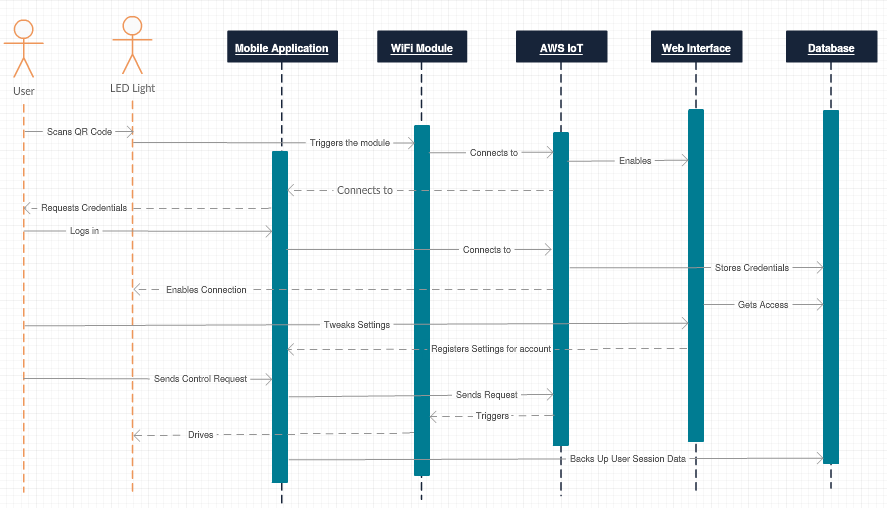
## Structural Modeling

It’s a design framework where the components, attributes and the relationships between them are identified and expressed. User being the driver class contains can own any number of devices which works in different modes of operation like scheduled mode and sensor mode. Sensor mode is a special case where the configured device works based on the outputs of the temperature and sound sensors. Scheduled mode is another way user controls the registered device with a preset schedule.



## Dynamic Modeling

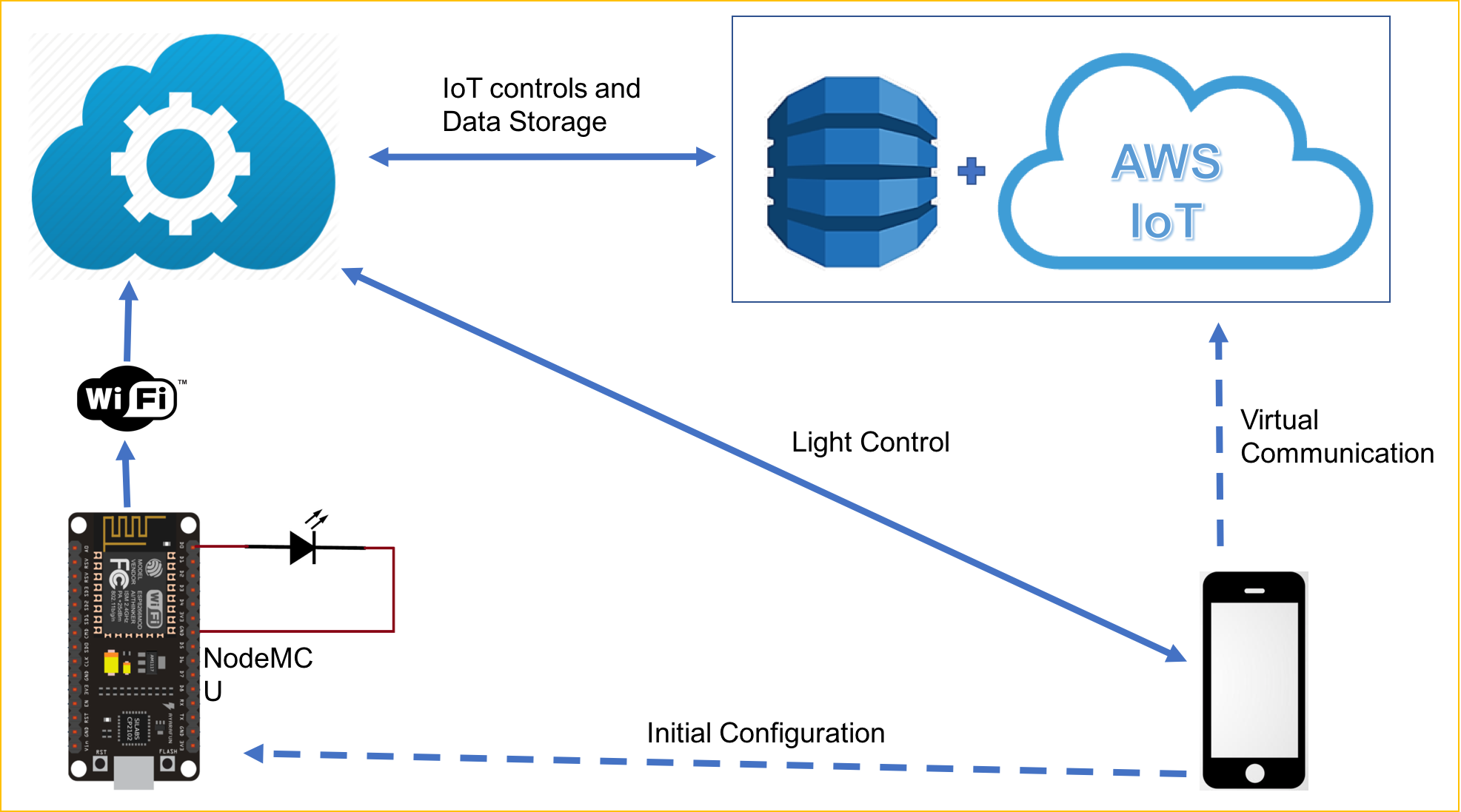
Behavior of the system which recurs over a period of time is defined in this modeling. Here the Sequence Diagram describes the action the system takes over time. As soon as the user registers the device, the system carries out a series of actions in an order. This is described in the figure below. The initial configuration of the Wi-Fi module with mobile applications is crux of the process. And the web interface takes the whole control after that which acts as a crux of communication gateway between mobile and IoT cloud, device and IoT cloud. Though IoT cloud and database are hosted on AWS as a single module its good practice to visualize them as two different entities to understand the sequence of service and dataflow better.

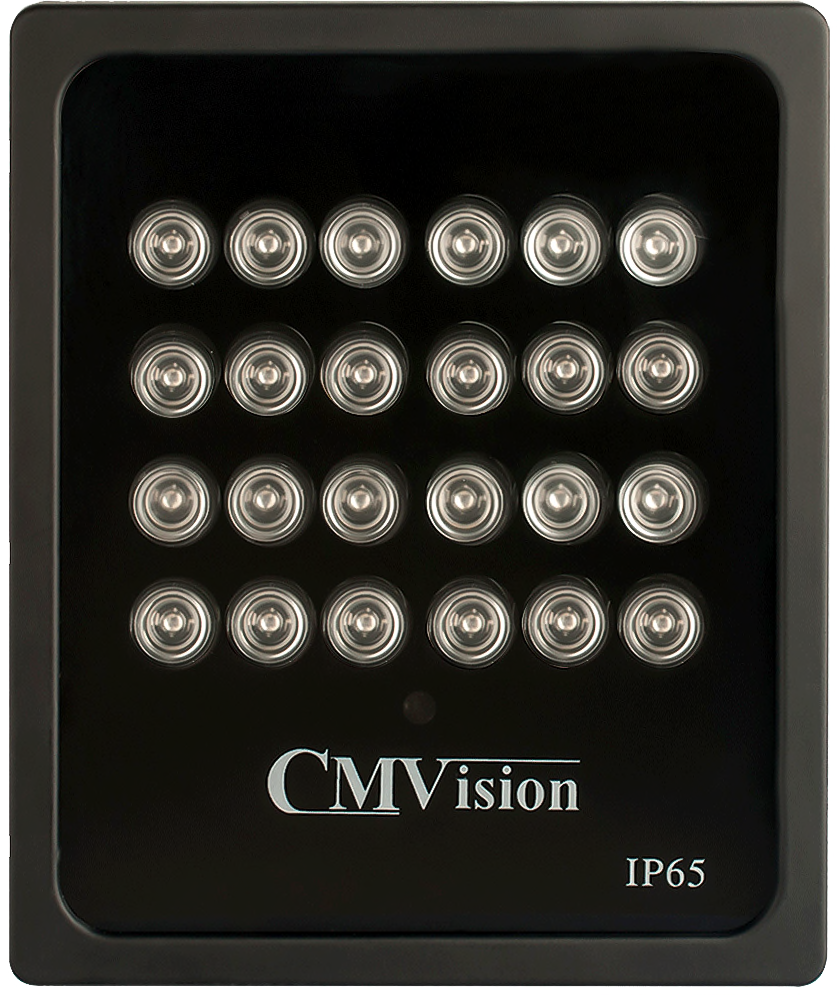


# Architecture

The architecture shows the building blocks of the whole system of which NodeMCU ESP8266 plays a key role, this is the processing side of the smart LED light that needs to be controlled. Controlling the NodeMCU ensures the functionality of controlling the smart LED light. Hence a web interface is used as a single point of contact to interact with the device. All other components are expected to communicate with this interface.

Mobile app, installed on a mobile device communicates with the web interface which is responsible to communicate with IoT cloud to store the data. And DynamoDB in IoT cloud is responsible for the whole control and storing the data, like all activities of the device, commands sent and received and to keep a track of activity. NodeMCU which can be mounted on to any device to enable Wi-Fi activity. Here we program it to control the intensities of the LED bulb.





We have used Amazon web services to primarily house a database in the EC2 server. EC stands for elastic cloud server. We use EC2 instance for the functionalities of webserver and MySQL server. HTTP server and MySQL server has been installed on the Linux instance on the EC2 server. Instance provides SSH and TCP for remote connectivity and for file transfer. We have configured the instance with PHP services which achieves the connectivity between the databases on the MySQL server and devices with HTTP request and responses.

In our solution MySQL server hosts two types of databases.

One of them is a master database with name caps18g6 which maintains all user details and device details and device user relationship. Each user owns a database which holds a command table and schedule table that receives commands to be executed and commands to be scheduled from the NodeMCU device. Hence the we made these database tables as a FIFO structure. Device makes PHP requests to request the last command that has been posted on the database table. Hence the commands that are executed are deleted which avoids the over consumption of the storage. We have developed the mobile app that posts PHP requests to the webserver to populate the commands on the user database. We have fulfilled the requirements to send the device to be illuminated in a given color combination of RGB. The device can be set with a schedule to switch on the device with white light and switch off the device at scheduled time. Intensity mode is designed to set the device intensity in white light mode with 1024 intensity levels.

## Database Design

Device (NodeMCU) side implementation:

We utilized the services of device, NodeMCU which is capable of connecting to Wi-Fi and has a capability of creating a hotspot. The ESP8266 is a 32-bit RISC processor with 160MHz clock speed. The 4K memory helps handle the parallel processes run relentlessly. NodeMCU has the micro controller unit ESP8266 which can be programmed and configured with different flashing soft wares. We have chosen Arduino which is widely used to program most of the programmable devices. And the programming language supported by the Arduino is C++. Hence the whole device side programming was done in the same programming language.

Libraries we have used for the development are ESP8266WiFi.h. This library helps in different Wi-Fi modes of connection. The device supports two modes of connection namely station mode and softAP mode. The first one helps in connecting the device with present Wi-Fi networks. Whereas softAP mode creates a Wi-Fi hotspot into which we can connect our mobile device for initial configuration. It helps in determining connection status of the device to the Wi-Fi. And as we know dealing with IP addresses would raise confusions we make use of named domain which needs use of DNS server implemented using DNSServer.h.

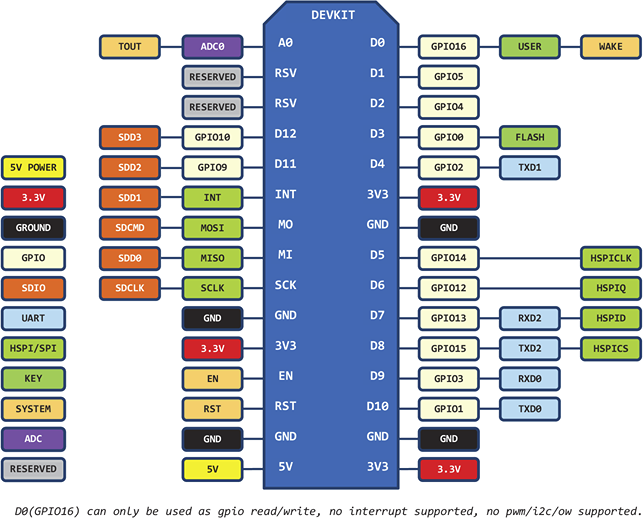
Communicating with device over a Wi-Fi connection has been implemented with help of HTTP request response. Hence a webserver needs to be deployed over the device which receives requests from other device and respond by publishing the webpages to the client. We have made use of ESP8266WebServer.h. Till this point we achieved a connection with in a local area network whereas the requirement needs the device to be controlled remotely which needs to connect with the AWS server to pull the request. For this we made use of ESP8266HTTPClient.h. we made use of client mode to make PHP requests over the internet. The URLs requested are briefly described in server-side development. Handling the response from the server by the device needs to get the information out of response. The response from the server side is configured to return in JSON format which is handled with help of ArduinoJson.h library.

Server-side implementation:

We have utilized Amazon Web Services to deploy webserver. Different services that helps control the devices are MySQL, HTTP server and PHP server which keeps running on the EC2 server restlessly to server the requests from the devices mapped to different users. The main way communication we chose to establish a connection between devices, mobile app and server is PHP requests. PHP scripts written on the server side get the parameters from mobile app and are used to database to update the commands and schedule. A list of PHP files is attached in the appendix along with the parameters. They can be classified into two types. We have designed few of them to pull requests from the device side. And Few of them are used to publish the commands from the mobile side. For example, getud.php was used to get the user mapped to the device id from the device side. So that the device can start pulling the commands that are meant for the particular device id.

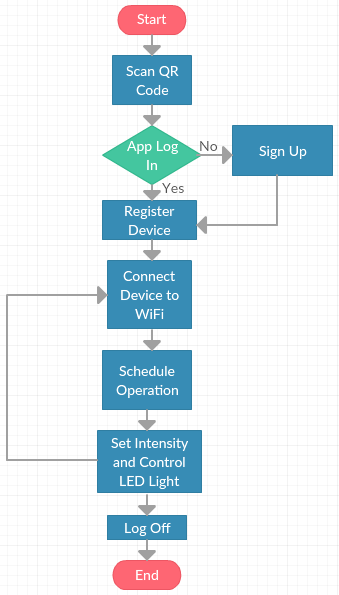
## NodeMCU Architecture

ESP8266EX is among the most integrated Wi-Fi chip in the industry; it integrates the antenna switches, power amplifier, low noise receive amplifier, filters, power management modules, it requires minimal external circuitry, and the entire solution, including front-end module, is designed to occupy minimal PCB area.



ESP8266 Pin Diagram

## User Interface Flow Chart



# Implementation

## Mobile Application

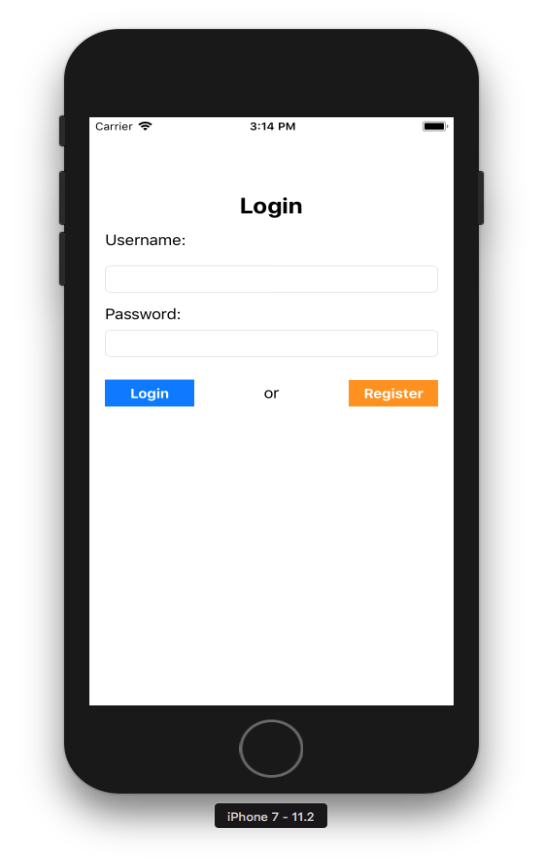
The user interface is created using the XCode IDE. XCode IDE provides a storyboard used to drag and drop all the components of the interface instead of writing the code. The application contains different views which are termed view controllers in XCode IDE. Each view controller provides different functionalities to the user and are connected to other view controllers based on the flow of application and the functionality of the components in that view controller. The view controllers are designed to meet all the requirement of the project like:

1. The user data must be secure.
2. The nodeMCU devices registered by a user can be accessed only by that user.
3. The user can add new nodeMCU devices.
4. The user can manually on/off the LED connected to nodeMCU device
5. The user can set the color and intensity of the LED connected to nodeMCU device.
6. The user can set the schedule for the LED on/off time.

The functionality of each view controller is described in detail below.

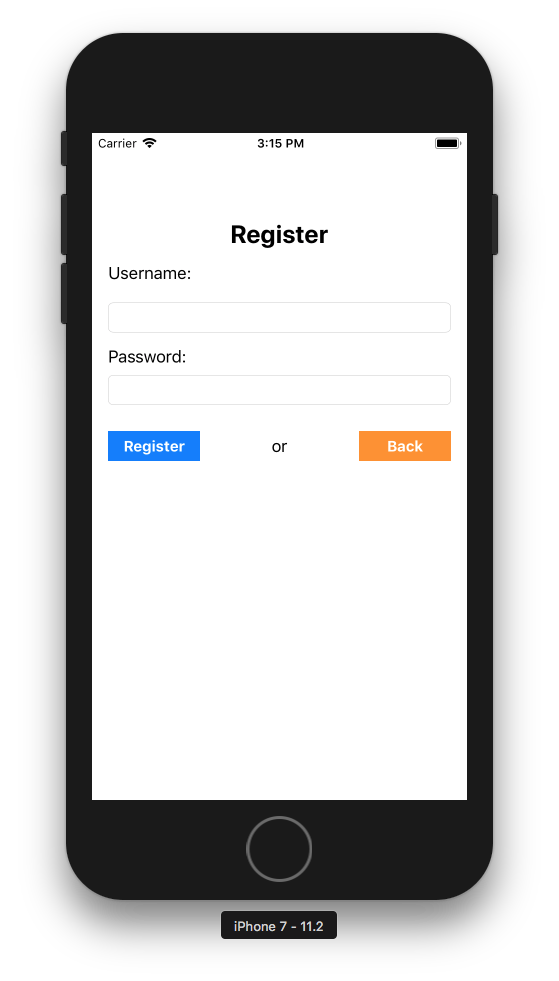
Login View Controller:

­



The login view controller has username text field, password text field, login button and register button. The user can login if he is already registered or can register by going to register page after clicking register button. When the user enters username and password and click login button, the username and password are sent to Users table present in the AWS to validate. The validation is posted from the validuser.php file. If the username and password fields are present in the Users table, a JSON object response is sent back to the login view controller. The JSON object contains validation is true or false and if validation is true the ID of the username is also retrieved. The view controller changes to Device view controller after successful validation along with the user ID and username. A label is displayed to check username or password if the validation retrieved is false.

Register View Controller:



The register view controller has username text field, password text field, register button and back button. The new user can enter username, password and click register. When the register but is pressed it checks for username and password text fields to be not null and. Then the registeruser.php action page is triggered to post username and password to Users table in AWS to save the user and a user ID is generated automatically in the Users table. Once the data is successfully saved in the table, a label is displayed on the view controller that the user has been successfully registered. If the user name is already present in the Users table, the label displays the username is already taken. The back button sends the register view controller back to login view controller, where the user can login with the registered username and password.

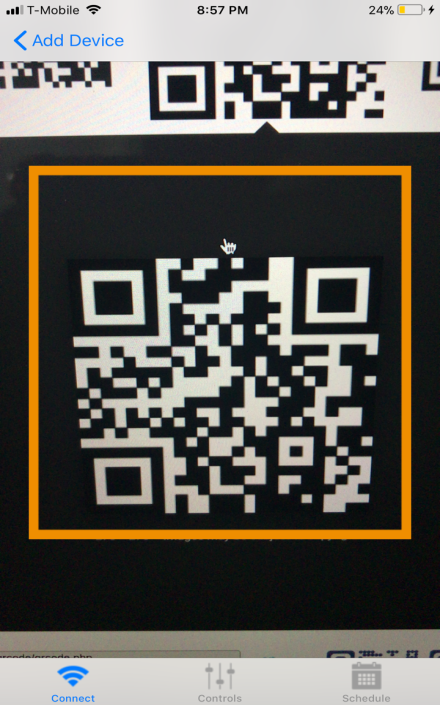
Device View Controller

The device view controller has table view, cancel button and camera button. When the user login, the view controller changes from login view to device view and in this view controller the username and user ID are retrieved from User Devices table in login view controller is sent to device view controller using a segue connected between the two view controllers. The table displays all the nodeMCU devices of the user. The nodeMCU device is connected to LED.



The nodeMCU devices are retrieved from User Devices table in AWS by running getud.php file which has the user ID of the present user. The cancel button acts as a logout action. The view controller goes back to login view controller from device view controller when the cancel button is clicked. New nodeMCU device can be added by clicking on the camera button present at the top right of the view controller. When the camera button is clicked the view, controller goes to Add Device view controller. When the device present in the table is clicked it goes to a tab bar view controller, it acts as an intermediate view between the device view controller and the next view controllers which are used to perform functionalities of the device connected to LED.

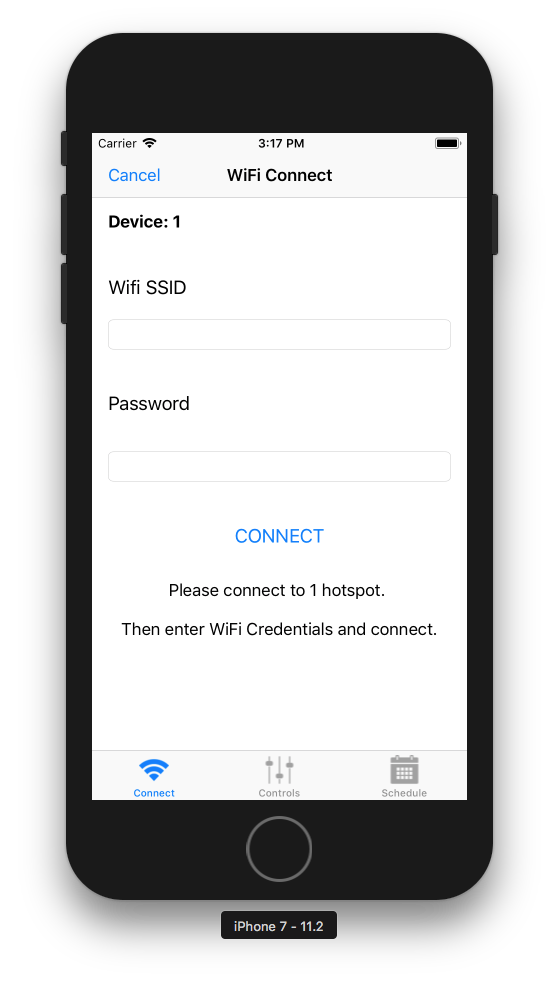
Add Device View Controller





The add device view controller opens the camera inside the application. The camera is used to scan the unique QR code of the device. When the QR code is scanned a popup is displayed. The popup displays the nodeMCU device ID and has options to retake the QR code, copy the device ID. When retake is selected, the user can rescan the QR code. When the copy option is selected the device, ID is posted to User Devices table in AWS by getud.php file. The device ID can be seen in the table in device view controller by selecting cancel button. The add device is one of the requirements of the project.

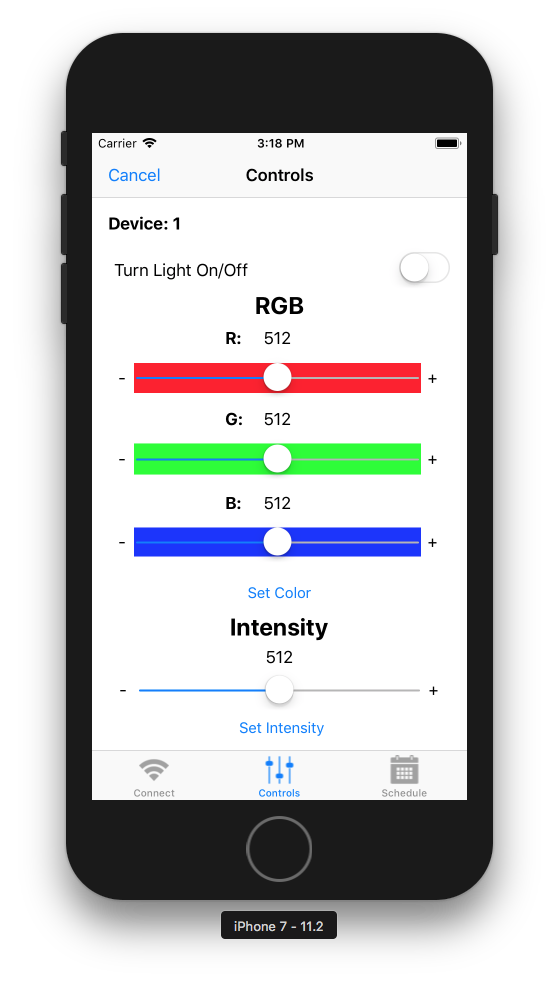
Connect View Controller



The connect view controller is the first view controller from the tab bar view controller. When a user selects the device from the table in device table view controller, the user ID, username and the device selected is sent to tab bar view controller which acts as an intermediate and sends this data to connect view controller and all other view controllers beyond the tab bar view controller. The connect view controller has Wi-Fi SSID text field, password text field, device label, notice label, connect button, cancel button and tab bar. The device label at the top show the device selected by the user from the table in device view controller. The cancel button presents at the top left side of the view controller acts as a back button and goes back to device view controller when clicked. The notice label at the bottom gives instructions to the user to connect to hotspot with the selected device as the hotspot name and its password. First the user connects to the hotspot of the nodeMCU device according to instructions. Then the local Wi-Fi SSID and password need to be provided by user in their respective text fields. When the connect button is clicked the Wi-Fi SSID and password are posted to Connect table in AWS by running connect.php file. The Wi-Fi SSID and password from the Connect table in AWS is sent to the nodeMCU device. After this the connection between the nodeMCU device and the mobile device is lost and now both are connected to the local Wi-Fi SSID provided. After this step the nodeMCU device is configuration is completed and the LED connected to the nodeMCU device can be accessed by the user from anywhere using this mobile application.

The tab bar at the bottom of the page is to toggle between the connect view controller, control view controller and schedule view controller. All the three view controllers are connected to tab bar view controller and can be selected from any of the three view controllers.

Control View Controller(RGB)

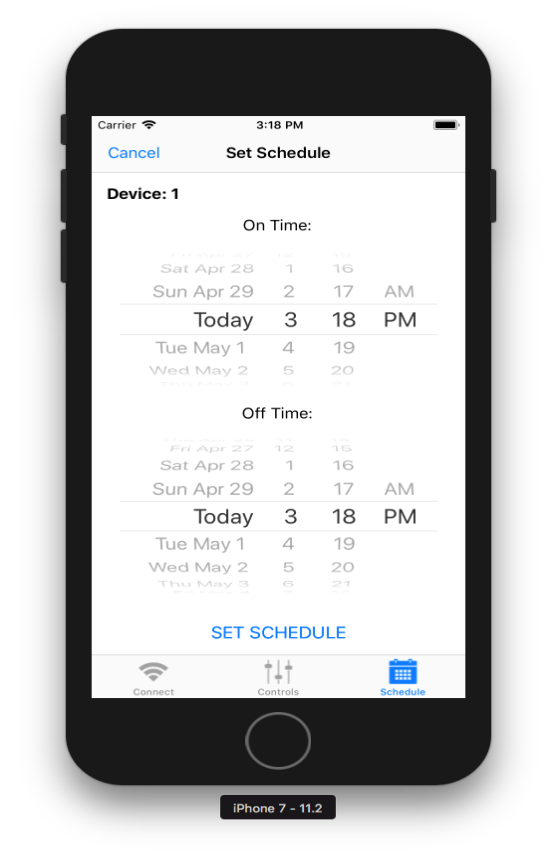


The control view controller is accessed when the device is selected in the device view controller and the view controller changes to tab bar view controller and then control tab bar button is clicked in the tab bar. The control view controller has device label, toggle switch, three sliders for RGB controls, and three labels for RGB values, et color button, intensity slider, intensity label, set intensity button, cancel button and tab bar. The device label displays the selected device from the table in device view controller. The toggle switch is used to manually on/off the LED connected to the selected nodeMCU device.

The manual mode to on/off the LED is one of the project requirements. When the toggle switch is set on a flag value of 1 and when is set off a flag value 0 is sent to Commands table in the AWS by running the usercommand.php file. The flag value from the Commands table is retrieved by nodeMCU device and sets the LED on/off. The RGB sliders are used to set the color of the LED and the RGB label displays each component value. When the RGB sliders are adjusted and set slider button is clicked, the RGB values are sent to Commands table in AWS by running the usercommand.php file. The RGB values from the Commands table are retrieved by the selected nodeMCU device and set the color of the connected LED light. Set the color of the LED is one of the requirements of the project.

The intensity slider can be adjusted, and its level is displayed by the intensity label present above the slider. When the set intensity button is clicked, the intensity level is stored in the Commands table in AWS by running the usercommand.php page. The intensity level is retrieved from Commands table in AWS is retrieved by the selected nodeMCU device and sets the intensity level of the LED connected. Set the LED intensity is another requirement of the project. The cancel button acts as a back button to device view controller. The tab bar at the bottom of the page is to toggle between the connect view controller, control view controller and schedule view controller.

Schedule View Controller



The schedule view controller is displayed after the nodeMCU device is selected from table in device view controller and schedule tab bar button is selected in tab bar. The schedule view controller has device label, on time date picker, off time date picker, set schedule button and tab bar.

The device label displays the selected device from the table in device view controller. The two date pickers set the on and off time of the LED connected to nodeMCU device. When the set schedule button is clicked, the on and off time is posted to Schedule Test table in AWS by running the schedtest.php file. The on and off time are retrieved by the nodeMCU device selected and set the LED according to the schedule. Set schedule is another requirement of the project. The cancel button acts as a back button to device view controller. The tab bar at the bottom of the page is to toggle between the connect view controller, control view controller and schedule view controller.

Tools and Software:

Arduino IDE:

Arduino IDE is open source software. It runs on Windows, Linux, macOS. The Arduino IDE makes it easy to write code on any Arduino boar and supports many languages. C++ language is used in this project.

AWS:

Amazon Web Services (AWS) provides on-demand cloud computing platform for developers. AWS gives many services like MySQL, Http Services, PHP Services.

MySQL Workbench:

MySQL Workbench is a database design tool. It is a visual design too. It supports database design, SQL development, creation, administration and design in the same application.

XCode IDE:

XCode is a software development tool for macOS by Apple to develop software for all its devices. XCode supports many languages and mainly popular languages among them are objective C and Swift. XCode has developer documentation and has a built-in simulator, a graphical interface of the real device.

Swift:

Swift is a programming language developed by Apple to develop applications for its devices. Swift supports many of the frameworks of the apple products with an extension to the objective C programming language.

## Web Interface

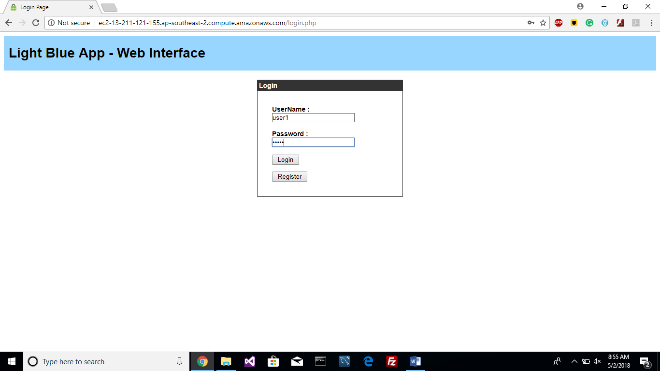
Beside the mobile app, our system has a web interface to control the light. So, any legitimate user can control a registered device from any remote place using the Web Interface. The Web Interface mainly consists of 6 php files.

Register.php

* Login.php
* Welcome.php
* Color.php
* Intensity.php
* Schedule.php

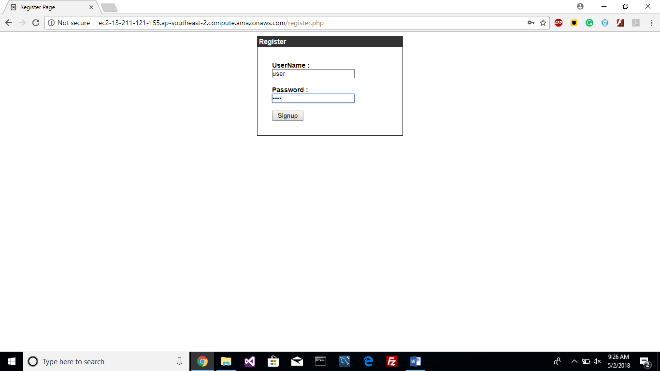
Login.php

The Login.php file takes the parameters Username and Password from itself. Then it validates for the given credentials from the users table in the caps18g6 database on the EC2 server. If the user was valid then it navigates to welcome.php else, it shows an error message.

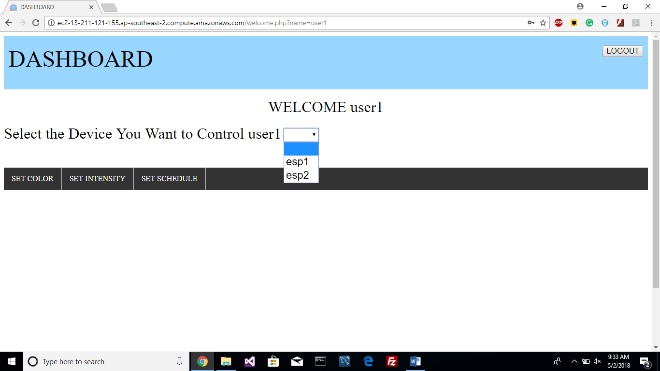


Register.php

The Register.php file allows the user to Sign-Up for a new account. This file takes username and Password as the input parameters and inserts them in to the users table in caps18g6 database on the EC2 server. It also further creates a database for a user with his name along with the tables commands and schedule test.



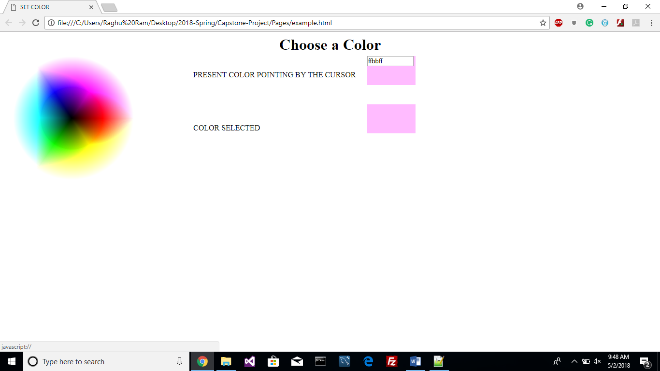
Welcome.php



After logging in to the system, it shows the list of the devices for the user who had logged in. It consists of a dashboard where the user has various options to control the light. It includes options for controlling the color, Intensity and Schedule the light.

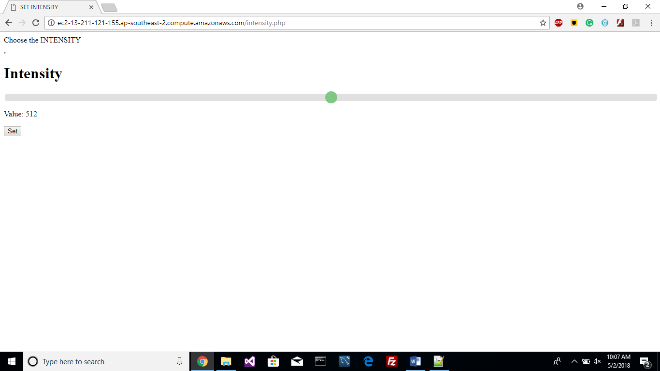
Color.php

The Color.php file is used to control the color of the light through Web Interface. It takes the RGB commands and stores these commands on to the command table which was in caps18g6 database with a command used for color.



Intensity.php

The Intensity.php file is used to control the intensity of the light through Web Interface. It takes the Intensity command and stores this command on to the command table which was in caps18g6 databse with a command used for intensity.



Schedule.php

The Schedule.php file is used to Schedule a session for the light through Web Interface. It takes the schedule command for both switching On and Off and stores this command on to the scheduletest table in the caps18g6 database.

## Steps

Server Side:

* We have created an Amazon Web Services instance with Linux OS on Elastic Cloud server.
* MySQL server, HTTP server and PHP services has been installed. The reference link helps in online document that helps in configuring the services on AWS EC2 instance.
* The services are configured to run automatically with start of the instance.
* PHP scripts can be included from the repository that has been submitted using FileZilla.
* The create statements have been submitted for the database creation and master database tables and user database tables.

Device Side:

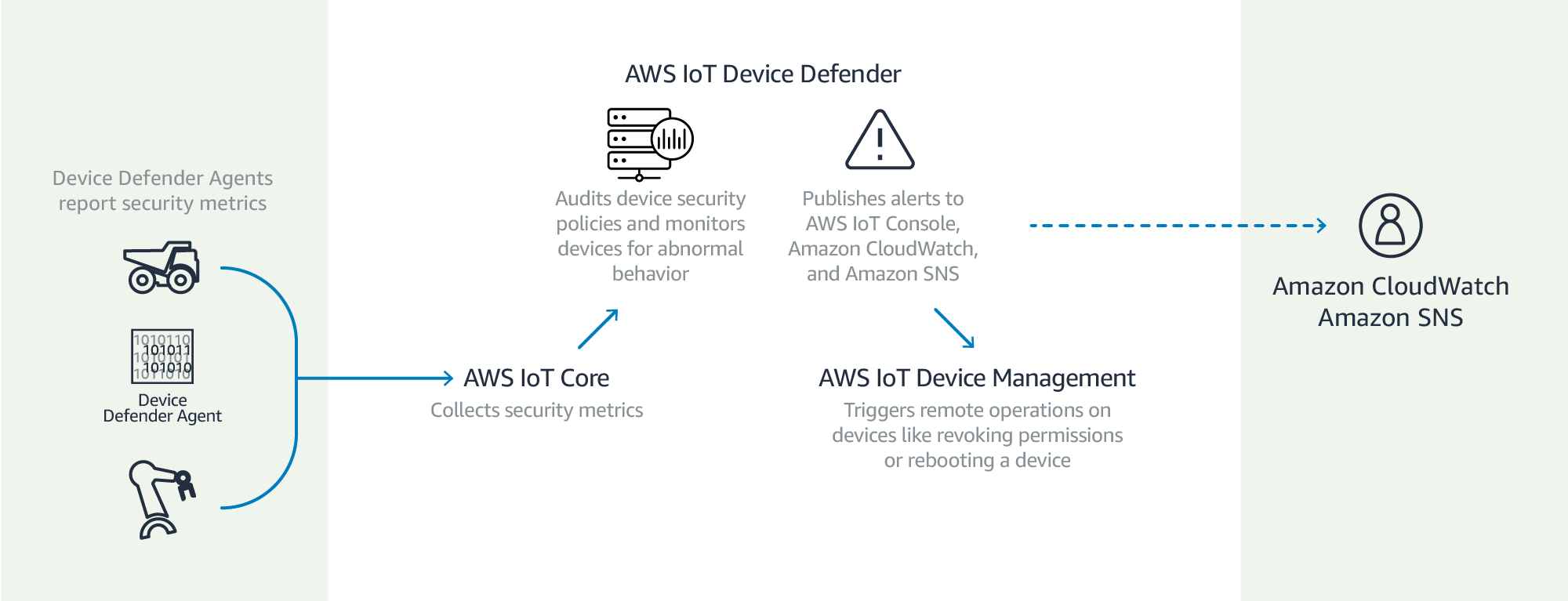
* Install Arduino latest version
* Use 115200 as upload frequency and 80MHz
* Select the right port and NodeMCU 0.9 board
* Use the given code and change diD in userCommand.h with unique number and flash the device.
* getUsername.h is used to identify the user to which the device is connected to.
* updateTime.h is used to synchronize the device.
* requestCommand.h is used to pull commands from the device side.
* rgbColors works as a main function.

# Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TEST | Memory Usage | Delay | Storage | Security | Stability |
| Mobile Application | 7 MB | < 3 Sec | 3 MB | QR Code | Good |
| NodeMCU | Nearly 4 KB | ~ 5 ms | 45 KB | Password | Excellent |
| AWS | < 1 GB | ~ 10 ms | 25 GB | Permission File | Excellent |
| Database | ……… | ~ 7 ms | …….. | User Authentication | Good |

# Future Work

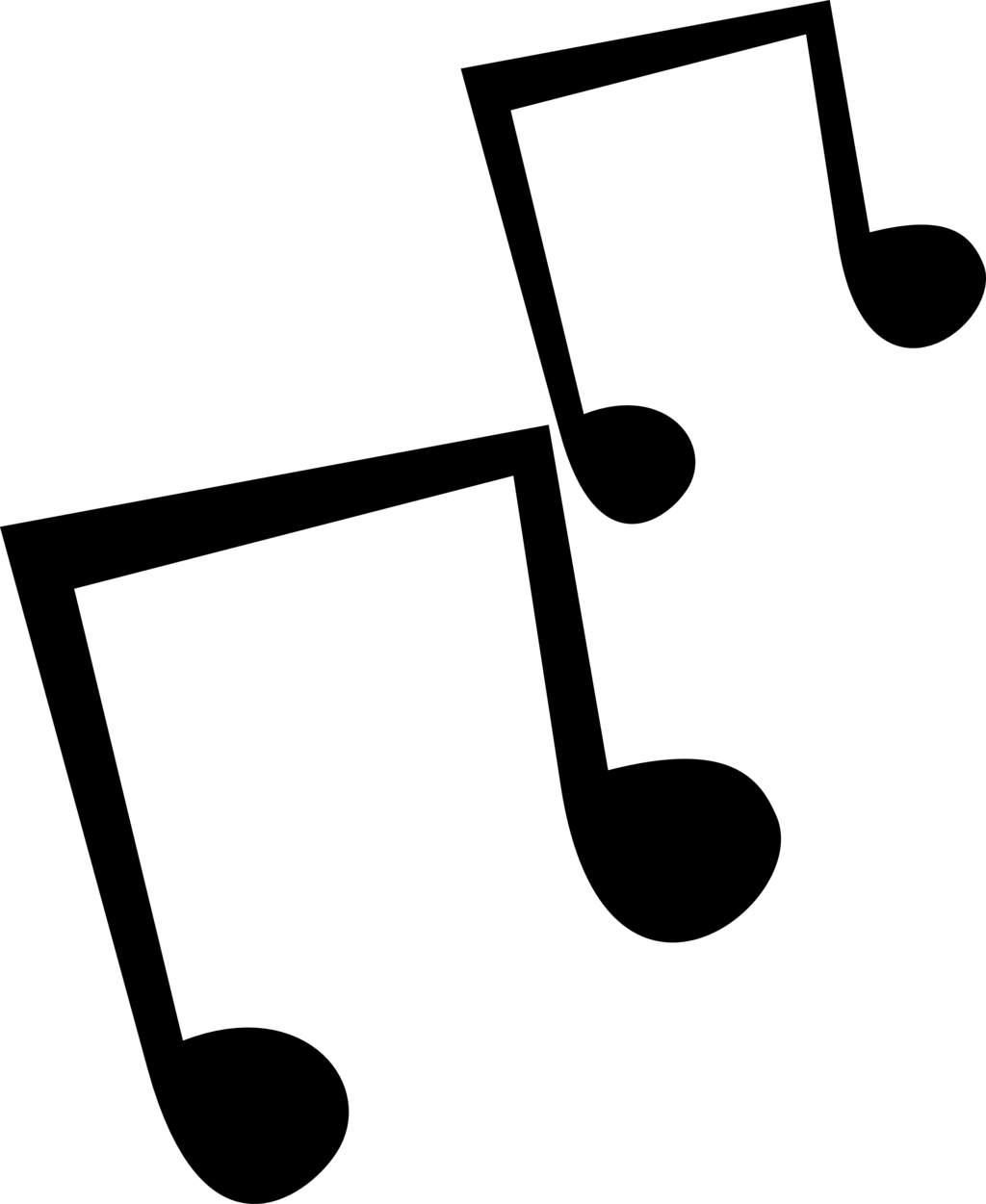
## Extended Cloud Support



Smart lights are targets for hackers because owners often use default passwords that come with their networks. We can use AWS IoT Device Defender to audit connection attempts to the device connected and if that data comes from an unauthorized endpoint, you will receive an alert.

## Added Functionality





Appendix

Appendixes, if needed, appear before the acknowledgment.

Acknowledgment

The preferred spelling of the word “acknowledgment” in

Music Controlled LEDs

For example, by adjusting its sensitivity we can make it less sensitive to reacts only on high note beats or we can also make it more sensitive to react on every little beat in the music.

Temperature Controlled LEDs

For example, if the reference voltage is 0.8V, then the voltage at the non-inverting input becomes 0.8V when the temperature is 80 degrees Celsius.

# Conclusion

* We successfully deployed the app with the required functionality using XCode, Arduino Workbench, MySQL DB and AWS Instance.
* AWS as a cloud service offers immense functionality though it is complex to setup when compared to a traditional network. Support for NodeMCU is limited.
* AWS could be further explored for device management and cloud security to work with a network of connected devices.
* NodeMCU is the most economical way to implement AWS IoT applications. It offers both basic and advanced programmable hardware functionality. Support for multiple platforms makes it flexible as well.
* MQTT is the primary protocol in play for communicating with AWS IoT server.
* MySQL DB is compatible with JSON format for commands. It works best with a server-based application.

# Team Member Responsibilities

*Sasidhar Pasupuleti*: Worked on hardware and AWS side deployment and integration.

*Sai Lochan Reddy Reddipalli*: Worked on design and development for web interface. Documentation and testing

*Raghuram Kambhampati*: Worked on database connections for user interface and hardware integration.

*Rohith Raj Reddy Alla Sai Poorna:* Worked on mobile application implementation and deployment.

References

1. Bertino, E., Choo, K. R., Georgakopolous, D., & Nepal, S. (2016). Internet of Things (IoT). *ACM Transactions on Internet Technology,* *16*(4), 1-7. doi:10.1145/3013520
2. P., S., S., S., M., A., & Ushaa, E. (2016). Realizing Internet of Things Using Arduino, ESP8266 and IIS Server and MySQL DB For Real-Time Monitoring and Controlling Multiple Fire Alarm Systems Over a Wireless TCP/IP Network. *I-manager’s Journal on Software Engineering,* *11*(2), 13. doi:10.26634/jse.11.2.13444
3. P, P. (n.d.). Beginners guide to ESP8266, from [https://tttapa.github.io/ESP8266/Chap04 - Microcontroller.html](https://tttapa.github.io/ESP8266/Chap04%20-%20Microcontroller.html)
4. Understanding the 802.11 Wireless LAN MAC frame format. (2017, April 26), from <https://witestlab.poly.edu/blog/802-11-wireless-lan-2/>
5. Kodali, R. K., & Soratkal, S. (2016). MQTT based home automation system using ESP8266. *2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*. doi:10.1109/r10-htc.2016.7906845

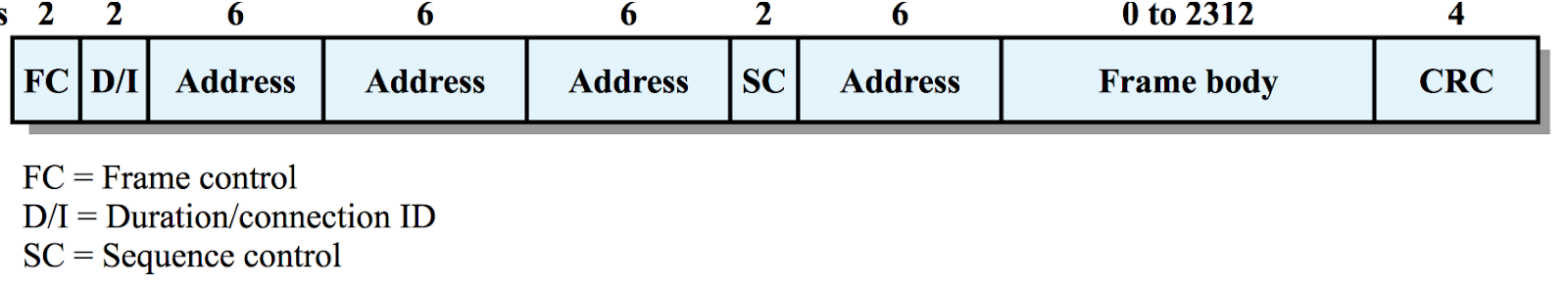
Appendix:

Protocol Specifications:

As we go through the TCP/IP stack for communication through the internet, we are going to deal mainly with the link layer and application layer rest of which is abstracted among these two layers.

**802.11 b/g/n**

  The Wi-Fi protocol is named as 802.11with different variations named as b, g and n which has different network strengths. A basic wireless LAN mac format looks as follows.

****

**Hypertext Transfer Protocol (HTTP)**

HTTP is an asymmetric request-response client-server protocol.  An HTTP client sends a request message to an HTTP server. The server, in turn, returns a response message.  In other words, HTTP is a pull protocol, the client pulls information from the server (instead of server pushes information down to the client). A URL (Uniform Resource Locator) is used to uniquely identify a resource over the web. URL has the following syntax:

protocol://hostname:port/path-and-file-name

There are 4 parts in a URL:

**Protocol**: The application-level protocol used by the client and server, e.g., HTTP, FTP, and telnet.

**Hostname**: The DNS domain name (e.g., www.nowhere123.com) or IP address (e.g., 192.128.1.2) of the server.

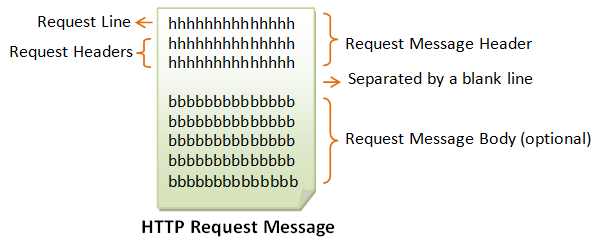
**Port**: The TCP port number that the server is listening for incoming requests from the clients.

**Path-and-file-name**: The name and location of the requested resource, under the server document base directory.

Examples of request line are:

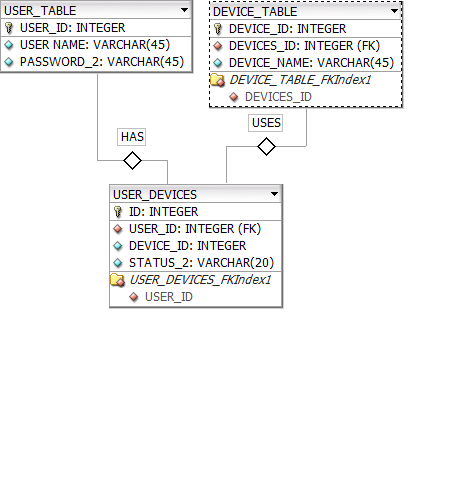
GET /test.html HTTP/1.1  
HEAD /query.html HTTP/1.0  
POST /index.html HTTP/1.1

The format of an HTTP request message:



The information is transferred in form of JSON format which are the key value pairs and can be handled easily with different programming languages.

Database and AWS:



1)Login.php

link:http://ec2-13-211-121-155.ap-southeast-2.compute.amazonaws.com/login.php

Description: This file posts the Username and Password that were taken from the Input fields. The script further validates the user from the caps18g6 database.

If the user is valid then it navigates to welcome.php file. Else it shows the error message.

2)Register.php

link: http://ec2-13-211-121-155.ap-southeast-2.compute.amazonaws.com/register.php

Description: This file takes the Username and Password from the Input fields. This file inserts a row in the database with the given user credentials. It also creates a database.

with the given username and creates commands and schedule test tables in the database having the name of Username.

3)Welcome.php

Link:http://ec2-13-211-121-155.ap-southeast-2.compute.amazonaws.com/welcome.php?name=user1

Description: This file gets username from the URL. Then this script shows the list of devices that were registered by user.

4)Color.php

Link: http://ec2-13-211-121-155.ap-southeast-2.compute.amazonaws.com/color.php?name=user1&device=esp1

Description: This file takes the parameters name, device, rvalue, gvalue and bvalue. This file inserts the RGB command on to the commnand table.

5)Intensity.php

Link: http://ec2-13-211-121-155.ap-southeast-2.compute.amazonaws.com/intensity.php?name=user1&device=esp1

Description: This file takes the parameters name, device and Intensity value. This file inserts the Intensity commands on to the command table.

6)SetSchedule:

Link: http://ec2-13-211-121-155.ap-southeast-2.compute.amazonaws.com/schedule.php?name=user1&device=esp1

Description: This file takes the parameters name, device and schedule times. This file inserts the schedule commands on to the schedule test table.

Source Code: https://github.com/caps18g6-Invictus

1. [↑](#footnote-ref-1)