**Build a Smart LED Light Mesh Network via Bluetooth**



**Team Members: Under the guidance**

Gaurav Kamireddi

Mohan Karigondanahalli Yiming Yang

Nikitha Naini &

Soumya Peruri Kewei Sha

**Acknowledgement**

We, Tricksters team, would like to express our gratitude to the project mentor **Mr. Yiming Yang**, for his valuable guidance, encouragement and timely suggestions

The successful accomplishment of this project would not have been possible, without the timely help and guidance rendered by **Dr. Sha**. We wish to express our sincere gratitude to Dr. Sha who have helped us to overcome challenges and for assisting us with project planning, paper reviews, presentation reviews and overall project management to complete the project.

Table of Contents

1. Motivation 4

2. Introduction 4

2.1 Overview 4

2.2 Acronyms and abbreviations 4

3. Requirements 5 4. Product delivered 5

5. Design of Project 5

5.1 Architecture Design 5

5.2 Database design 9

5.3 Protocol/ Algorithm design 9

5.4 Flow Design 11

6. System Implementation 12

7. Testing 14

8. Conclusion 14

9. References 15

10. Appendices 15

# 

# 

**1.Motivation**

What if you are standing at the pavilion and want to switch on lights on the other end of the stadium? Would you prefer going in person to a set of switches located on the other end or would you consider switching them on from the pavilion through a Web User Interface?

Bluetooth Low Energy (BLE) supports a mesh topology to establish many-to-many (m:m) device communications. The BLE mesh capability is optimized for creating large-scale device networks and is ideal for building automation, sensor network, and asset tracking solutions.

Using the BLE mesh networking technology, we can connect each light via Bluetooth and control them from a centralized location through a Web User Interface.

# 2.Introduction

## 2.1 Overview:

This project is to build Smart LED light mesh network using Bluetooth low energy (BLE) 4.0 channel available on Raspberry Pi3 module. The light mesh is created using self-organized algorithm. Web User interface is developed to display the mesh topology of lights to allow User to view the status of light in the topology and to perform the operations(ON/OFF) on them.

**Scope:**

* Design and implementation of efficient algorithm to mesh topology of Smart LED lights.
* Design and Development of Web UI to display the Smart LED lights mesh topology.
* Design and Development of BLE (Bluetooth Low Energy) 4.0 communication protocol to enable communication between Smart LED lights.
* Design and Development of control of operations (Status and ON/OFF) on individual Smart LED lights from Web UI.

## 

### 2.2 Acronyms and abbreviations

|  |  |
| --- | --- |
| **Acronym or**  **abbreviation** | **Definitions** |
| BLE | Bluetooth Low Energy |
| UI | User Interface |
| GUI | Graphical User Interface |
| Pi3 | Raspberry Pi version 3.0 |

**3.Requirements:**

**Functional Requirements:**

* Smart LED light should be able to inquire the other Smart LED lights in range.
* Smart LED light should be able to connect to other to send operations command
* Web UI should be able to send inquire command to Smart LED Light via BLE 4.0.
* Web UI should be able to receive response for inquire command from the Smart LED lights.
* Web UI should be able to create mesh topology of Smart LED lights and display the same.
* Smart LED light should be able to respond to the inquire command received from Web UI.
* Web UI should be able to send operations (Status and ON/OFF) command to Smart LED lights via BLE 4.0.
* Smart LED lights should be able to receive operations (Status and ON/OFF) command from Web UI via BLE 4.0 and operate accordingly.

**4. Product Delivered**

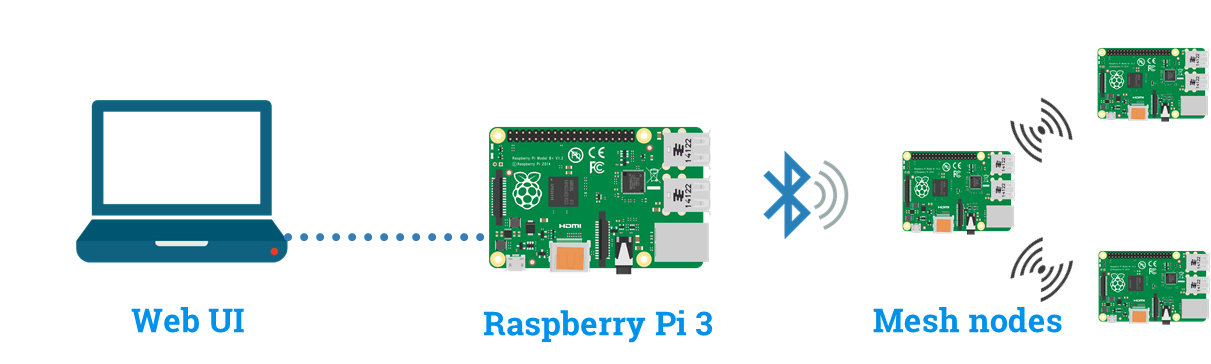
Smart LED lights mesh created using Pi3 modules using Bluetooth Low Energy 4.0. Smart LED light mesh topology is displayed and available on Web User interface for user to control its operations(ON/OFF) and view status of each remotely.

**5. Design Phase**

**5.1 Architecture Design:**

Blue Mesh system mainly holds three components:

* Web User Interface
* Raspberry Pi3 as a server
* Bluetooth Mesh network(Pi3’s)



**Name :** Raspberry Pi3/BLE010V5

**UUID :** EE:52:5B:04:00:02

**Frequency Range :** 2018MHz

* **Web User Interface:**

This section presents the GUI of Blue Mesh application which is easy to access and provides a real-time visualization of the LED lights as a mesh network. Logical view of all the functionalities are displayed according to the entity type. Every element is exhibited as a separate control and are selected by appropriate tabs. Web user interface allows user to discover the mesh topology of the Pi3 Bluetooth module in the network. User will be able to click on the ‘Discover’ button to allow the web UI to trigger the discover request to the web server for collecting the BLE mesh topology to display the virtual topology of the mesh for the User.

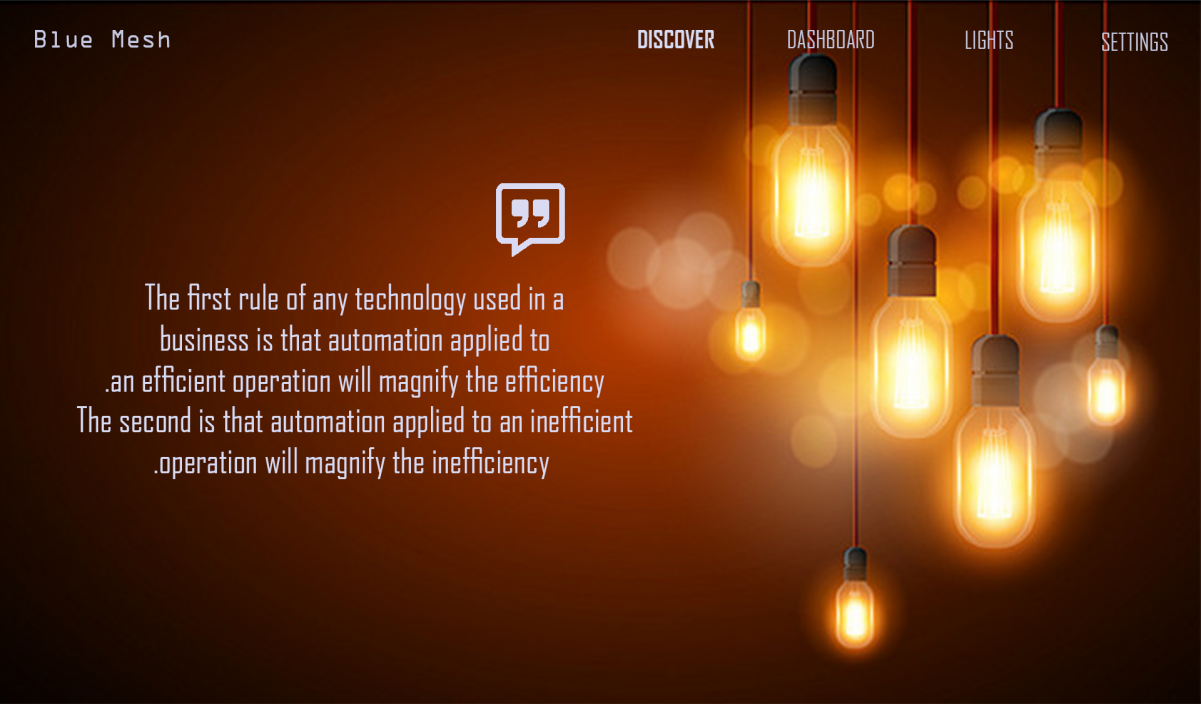


Figure 1: Web UI before Discovery

Web User interface incorporates three methods to perform the necessary operation for the User:

Discovery ()

Control ()

MeshBuild ()

**Discovery ()**: This method will capture the User request for discovery when the User click on the ‘Discovery’ button on the UI. And then the request is forwarded to the webserver to discover the BLE mesh topology available in the network.

**Control ()**: This method will capture the control operations ON/OFF from UI when User choose a light available in the mesh displayed and performs action ON/OFF and the same control is forwarded to the webserver to communicate the requested BLE node in the mesh network.

**MeshBuild ()**: This method will create and display the mesh topology in Web UI of the BLE based on the mesh list that were captured by the webserver by discovery operation.



Figure 2: Web UI after discovery of BLE mesh

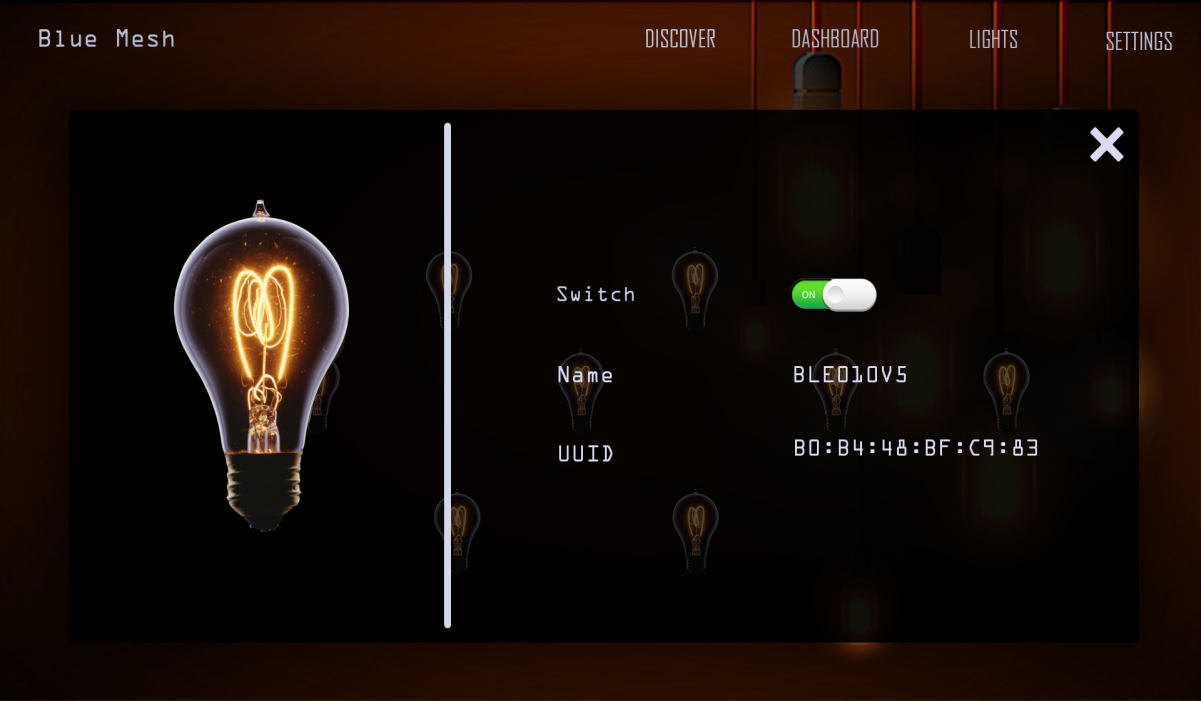


Figure 3: Control operation on Web UI

* **Web Server:**

It is the latest version of credit card sized computer from the Raspberry Pi foundation. It can interact with the outside world and has been used in wide array of projects. The Bluetooth module in Pi is used to interact with another Pi3 module which is having the Bluetooth functionality. Node.js installed on the Pi provides a server interface. The Raspberry Pi 3 will be running the web server to enable communication between the Pi3 mesh nodes and Web UI. The web server receives requests from web UI for following operations:

Discovery ()

Control ()

MeshStorage ()

Path ()

**discovery ()**: This operation in web server will receive the discover request from the Web UI and perform the scan operation through the underlying Bluetooth module in it to discover the all the available Pi3 Bluetooth modules in the network.

**control ()**: This operation in web server will receive the control operation (ON/OFF) and the Bluetooth node details on which the operation need to be performed from web UI and communication the same to the requested node using the forwarder list in the bluetooth payload.

**MeshStorage ()**: This operation will store the mesh list available in the response from the Pi3 Bluetooth modules during discovery operation. The mesh details will be stored in the xml format.

**path ()**: This operation in web server will create the optimum path for the destination node on which the control operation need to be performed and add them in to the forwarder list for the control () operation to forward it to the mesh network.

* **Raspberry Pi3 as Client:**

For developing a mesh network, we are using four Raspberry Pi3 devices. More than two devices are used to create a mesh network. All the devices are self-organized. This module is connected to Raspberry Pi3 Server via Bluetooth module and extracts the information of control operation and performs accordingly. Every module is controlled individually in a network of mesh. Flooding algorithm is used to provide connection to every device which are even at the distant proximity.

The Pi3 client module performs following operations:

**Advertise** () - This operation is done by each device to advertise about their availability.

**Connect** () - This operation will control the required device by reading the forwarded node list.

There are two services that Pi3 would use to enable to create mesh and forward the control operations to the destination node in mesh topology: GATT service, GAP Service

GATT service is also known as Generic Attribute Protocol(ATT) which is used to discover the bluetooth nodes in the mesh, to identify services and store, characteristics and related data in a simple lookup table.

**5.2 Database Design:**

In this project we will be using xml to store the data. The data that will be stored are the details of the Pi3 bluetooth mesh nodes and their neighborhood nodes. The neighborhood nodes are stored as MeshList under each Pi3 node which will be used for building mesh topology in Web UI and to determine the path while sending the control operations to the destination node.

*<PiNode>*

*<NodeUUID> - - - </NodeUUID>*

*<Status> On / Off </Status>*

*<MeshListUUID> - - - </MeshListUUID>*

*- - -*

*- - -*

*<MeshListUUID> - - - </MeshListUUID>*

*</PiNode>*

Above xml shows the format of the details stored for each Pi3 node that is discovered in the mesh network.

**5.3 Protocol/Algorithm Design:**

In this section the design of Protocol/Algorithm for discovering the Pi3 Bluetooth modules in the mesh topology and also communicating the controls to the desired node to perform the ON/OFF operations are discussed.

**Discover Mesh:**

The Raspberry Pi 3 will behave as a web server. It would be running the Node.js as web server which will also provide the GATT and GAP service to enable discovery of Pi3 client nodes in its vicinity.

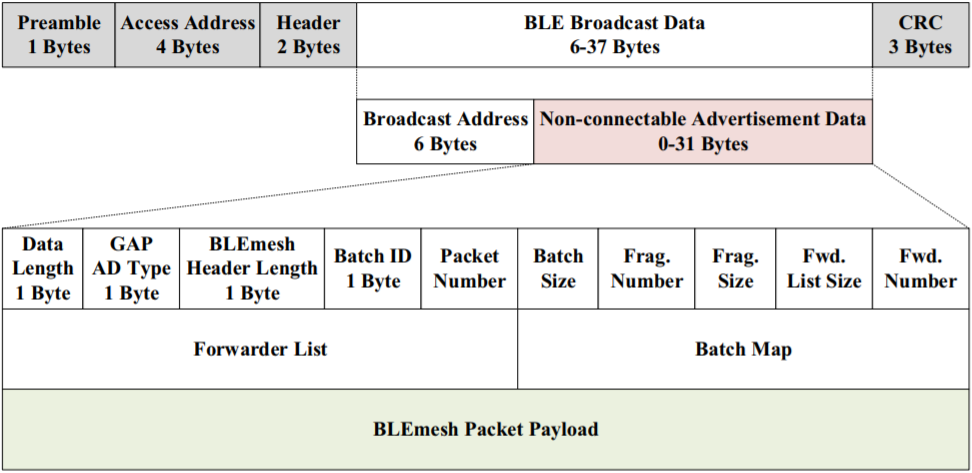


Figure 4: BLE packet

The Figure 4, shows the components of the BLE packet that will be used to transmit the data between the Bluetooth nodes in the mesh topology. The Broadcast data will be sent by the gateway in the BLE Broadcast Data segment of the packet. The response from the Pi3 client nodes can be send in the BLEmesh packet payload.

|  |  |
| --- | --- |
| Name | Pi3 |
| UUID | B8:B8:27:EB:23:E2:A4 |
| Mesh List | UUID1, UUID2 |

Table 1: Bluetooth payload

In the able Table 1, when the Raspberry Pi(Gateway) starts scanning for Pi3 Client Bluetooth nodes in its vicinity the nodes will respond back to the gateway with its discovery of neighboring nodes as well and create a Mesh List for the immediate nodes that are discovered as shown above and send it back to the gateway. Using this MeshList the gateway will be able to provide the Mesh topology to the Web UI to display the same to the User to perform the control Operation.

**Control Operation:**

The Web UI will be able to send the control operation from the User input on the desired BLE node from the mesh topology displayed. The User will select the node fro the UI and select the ON/OFF operation and the operation will be triggered to the web server. The web server will find the path to the node using the MeshList that is available in the database.

|  |  |
| --- | --- |
| UUID3: B0:B4:48:BF:C9:83 | Priority 1 |
| UUID2: B0:B4:48:ED:44:C3 | Priority 2 |
| UUID0: B0:B4:48:GA:F3:A1 | Priority 3 |

Table 2: Forwarder list from BLE pay load

The web server will find the path of the destination node and add them into the forwarder list. The destination node is given highest priority and the source node is given the lowest. Based on the priority the nodes will forward the control operation to the destination node. From the Table 2 we can see that the UUID3 is the destination which is assigned with highest priority and the source node UUID0 is assigned with lowest priority.

**5.4 Flow Design:**

In this section the detailed flow of the project is described as a flow design:

**Discovery Flow:**

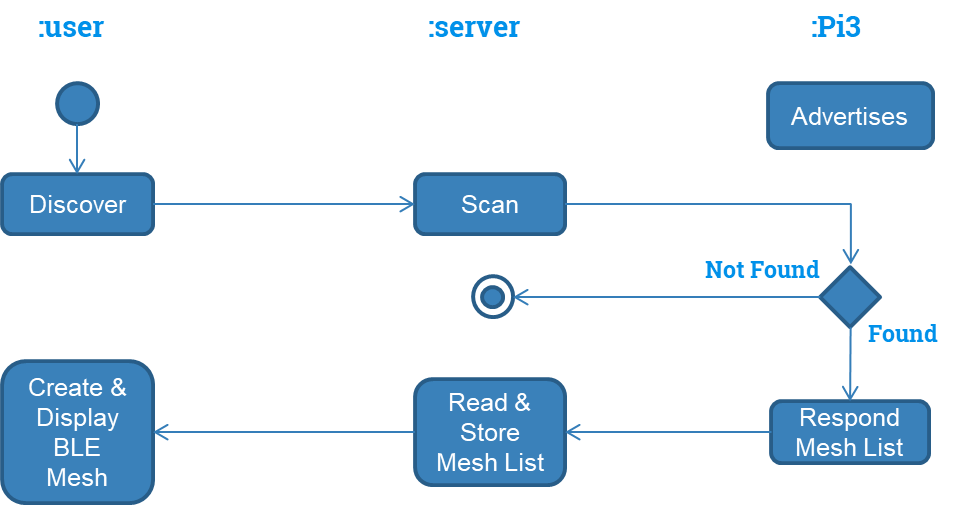


Fig: Flow of discover operation

This gives the flow of discovery operation that is used by the user to have control over the mesh of lights. Pi3 mesh modules will always be in advertisement state. When a user clicks discover button, the server receives it and scans for the mesh devices availability. If found it responds by providing the mesh list to the server which in turn store the list and displays mesh on the UI.

**Control Flow:**

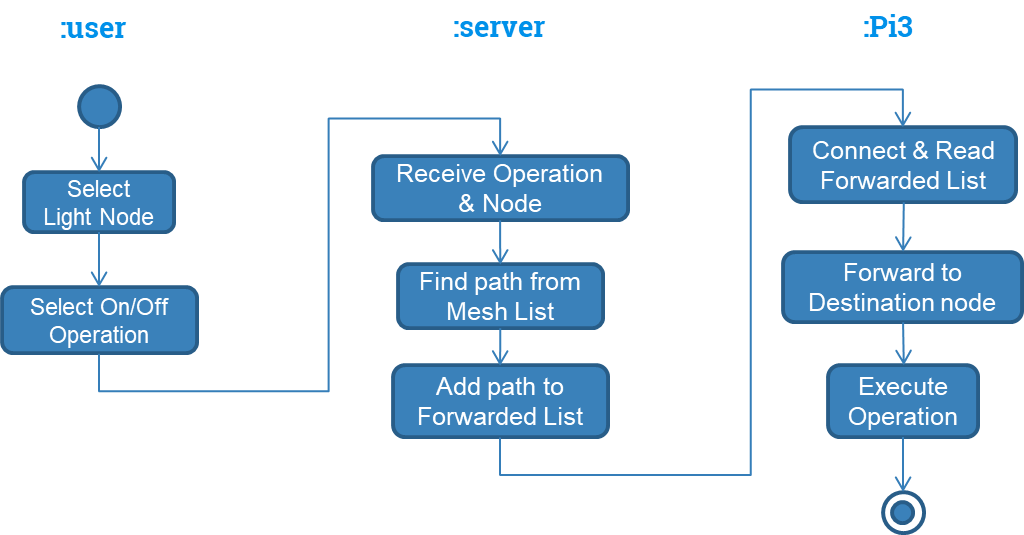
****

Fig: Flow design of Control Operation

This is the flow for Control operations. After the discovery process the UI will provide interface to select the device and the operation need to be performed. Once it is done the server receive the data and finds the path accordingly and adds it to the forwarded list through which the Client Pi3 devises find the destination node and executes the operation.

**6.System implementation**

System implementation phase was done in certain steps: Gathering design requirements, server establishment, Mesh network among 4 devices and website development.

**6.1 Design Requirements:**

* **Software Requirements:**
  + Operating system: Windows 10, Linux
  + Programming languages: C, Node.js, Angular.js
  + Tools: Visual Studio 2017
* **Hardware Requirements:**
* Device: Raspberry Pi
* Version: 3.0

**6.2 Discovering Mesh network:**

* Angular.js technology is used for building the UI.

When one requests to discover the mesh of lights available in the network, the UI communicates with the server to get the list of nodes in mesh. The server scans the nearby mesh nodes which in turn scans for the other available mesh nodes and stores the list of nodes as an array and sends the complete list of nodes to the server which stores this list in XML(database).

**6.3 Controlling the lights:**

**Web development:** When a user wants to control a light in a mesh network, the UI is designed to send the control request to the server Pi to perform the operation. The status and details of a node in mesh are made available in UI from the database through server.

**Server establishment:** Raspberry Pi-3 with inbuilt bluetooth and WIFI module is used as a server that stands as a gateway between mesh network and web UI. Raspbian OS is installed on it and is set up with Node.js. Code for 1) scanning and connecting the nearby bluetooth device using Node.js 2) taking the request from UI 3) passing the request to the connected device in mesh and 4) passing the response back to UI is developed. Server receives the request from UI and starts communicating/requesting the nearby mesh device.

**Mesh network:** Four raspberry Pi devices that are setup and installed with noble and bleno bluetooth packages, are used to create a mesh of Led lights. Each Pi is put to be in advertising state before we perform any operation. Every node in the mesh acts as both server and client depending on the operation. When the server receives the request from UI, it scans for nearby mesh device and passes the request to it. This mesh device now changes its state from client to server and starts scanning other nearby devices to pass this request and, every other device continue this procedure until it reaches the required destination node/device. The destination node then performs the action according to the request.

**7. Testing**

* **Positive Test Cases:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Description** | **Expected Value** | **Actual Value** | **Result** |
| **1.** | Open the application | Go to main page | Opens main page | Pass |
| **2.** | Click on discover | Displays mesh of lights | Mesh of lights displayed | Pass |
| **3.** | Click on any light | Displays light details | Light details displayed | Pass |
| **4.** | Turn on light | Respective light should be turned on | Light turned on | Pass |

* **Negative Test Cases:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Description** | **Expected Value** | **Actual Value** | **Result** |
| 1. | When we turn on a specific light, other lights should not change their state. | Only operated light changes its state | Only operated light changed its state | Pass |

**8. Future work**

* Grouping two or more lights and be able to control them.
* Adding motion sensor hardware features and applying them to switch on the lights automatically.
* Changing intensity of the light to control power consumption.
* Adding color changing facilities.
* Scheduling the light to switch on for a particular duration.

**9. Conclusion**

With the development of Bluetooth LED light mesh network, one can operate any light in the network from a Web interface that supports bluetooth technology. Virtual display of mesh network in the web UI gives easy access to the lights in the mesh. Use of bluetooth technology over WIFI results in low cost, low energy product development.

**10.References**

1. <http://eeca2.sogang.ac.kr/publications/international/BLEmesh_A%20Wireless%20Mesh%20Network%20Protocol%20for%20Bluetooth%20Low%20Energy%20Devices.pdf>
2. <https://blog.bluetooth.com/introducing-bluetooth-mesh-networking>
3. <https://ieeexplore.ieee.org/document/7370451/>
4. <http://visjs.org/network_examples.html>

**11. Appendices**

**Appendix A: Project timeline**

The project duration was ~14 weeks, starting from Jan 23, 2018 and ending on May 4, 2018. Based

on the complexity of the task, time is assigned to each task.

* **Requirement analysis**: Feb 6 - Feb 13

Took a week to gather all the requirements needed for the project.

* **Design Analysis**: Feb 13- Feb 20

In this stage we designed complete architecture of the project, flow design and database design.

* **Raspberry Pi 3 Connection**: Feb 20- Mar 13

Connected Raspberry Pi 3 as a Web Server and able to send and receive messages with the server.

* **Web User Interface**: Mar 13-Apr 17

Design of the Web User Interface to connect with the Raspberry Pi 3 server and the mesh topology.

* **Implementation**: Mar 13- May 1

After setting up the Raspberry Pi3 as web server, it was successful in discovering the nearby mesh nodes and establishing the communication between the server and mesh nodes and the mesh nodes itself.

* **Final Demonstration**: May 4

Final demonstration of the Smart LED Bluetooth Mesh Network.

**Appendix B: Responsibilities** s

|  |  |
| --- | --- |
| **Tasks** | **Responsible Team Membe**r |
| Web UI development | Nikitha, Soumya |
| Server-side development | Mohan, Pratik, Soumya, Nikitha |
| Mesh topology | Mohan, Nikitha |
| Presentations | Pratik, Soumya |
| Raspberry Pi setup | Soumya, Nikitha |
| Technical Report | Soumya, Pratik |

**Appendix C:**

**Document Version log:**

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Version** |
| 02/06/2018 | Mohan Karigondanahalli Chandrashekar Murthy | Draft 1.0 |
| 02/27/2018 | Pratik Kamireddi | Draft 1.2 |
| 03/15/2018 | Nikitha Reddy Naini | Draft 1.3 |
| 4/1/2018 | Soumya Peruri | Draft 1.4 |
| 4/20/2018 | Soumya Peruri | Draft 1.5 |
| 4/29/2018 | Soumya, Nikitha, | Draft 1.6 |
| 5/3/2018 | Soumya, Pratik, Nikitha, Mohan | Draft 2.0 |