# ADAPTIVE AND SELF-LEARNING SMART STREET LIGHTING AUTOMATION

## USER MANUAL

THE VIGYAANEERS

Paras Lehana

Ayushmaan Bansal

Yash Garg

Abhinav Bansal

RAJASTHAN DIGIFEST BIKANER

25-27 July 2018

**July – 2018**

# TABLE OF CONTENTS

## **SUMMARY**

Adaptive Smart Street Lightning is a module to automate existing street lights infrastructure using deployment of cost-efficient ESP8266 WiFi chipsets. The module optimizes group's net intensity (thereby energy costs) using self-learning over attributes like area light intensity (using LDR), weather conditions (like cloudiness, temperature, visibility), geolocation (using Google Maps API), time of the day and Master Control. A group of Street Lights (belonging to a colony or area) is a (sub)network and thus bears an IP address that can be controlled by the administrators using local internet/Wi-Fi (the android interface that allows this is the Master Control). Moreover, the street network is not hard-coded into the app but is managed via cloud server (Firebase) allowing real-time management. The module will thus help connect a network of street lights arrive at a collective optimized intensity to further reduce net energy consumptions as well as collect environmental parameters that could be used to generate datasets for statistical analysis and predictions using Fuzzy Inference System and/or Machine Learning Algorithms. The approach can be extended to whole city or state department and can additionally help rectify local errors such as a non-working street light or those depicted by the collected data. While administrators would have advanced tools to control and manage the network through mobile or web authorized interfaces, citizens would be able to view their local and global street lights cluster and can have further access to corresponding open data.

# CHAPTER 1: ESP8266

## 1.1 Download Drivers, Flasher and IDE

1. Download and install CP2101 driver
   https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

2. Download NodeMCU firmware flasher (EXE)
   https://github.com/nodemcu/nodemcu-flasher

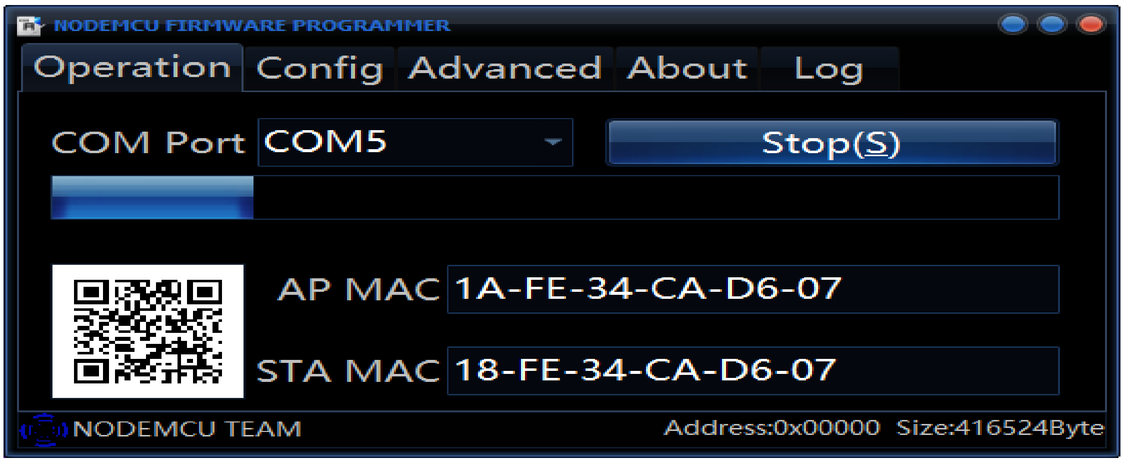
3. Download ESPlorer IDE (JAR)
   https://github.com/4refr0nt/ESPlorer

## 1.2 Flashing NodeMCU firmware

1. Open ESP8266Flasher.exe (downloaded in 1.1.2).

2. Connect ESP8266 to one of the USB port.

3. Select this USB port in the dropdown menu for COM Port in the flasher. The COM Port code (e.g. COM5) can also be viewed in the Device Manager.



COM Port COM5

AP MAC 1A-FE-34-CA-D6-07

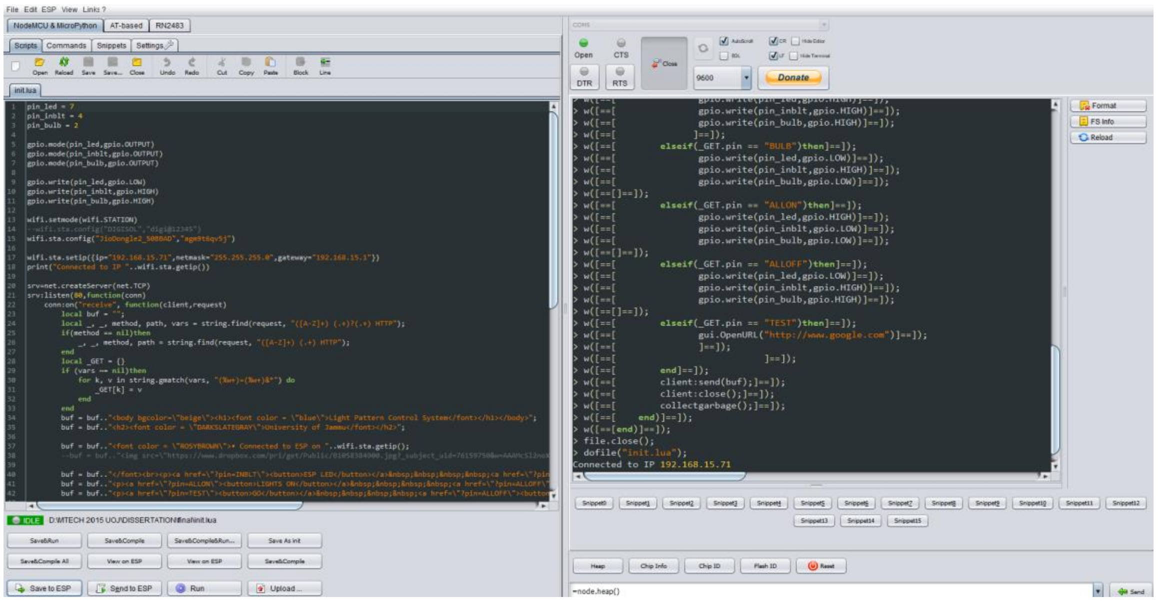
STA MAC 18-FE-34-CA-D6-07

Address:0x00000 Size:416524Byte

4. Note down these MAC addresses. AP MAC refers to the MAC address of the device when it works as an Access Point while STA MAC is used when the device acts as a station which is connected to some other AP. This project uses the device as a station which is connected to the local AP of the street area.

## 1.3 Uploading LUA code to ESP8266

1. Open ESPlorer (downloaded in 1.1.3). You will need JRE for running this JAR file.



```
pin_led = 7
pin_inblt = 4
pin_bulb = 2

gpio.mode(pin_led,gpio.OUTPUT)
gpio.mode(pin_inblt,gpio.OUTPUT)
gpio.mode(pin_bulb,gpio.OUTPUT)

gpio.write(pin_led,gpio.LOW)
gpio.write(pin_inblt,gpio.HIGH)
gpio.write(pin_bulb,gpio.HIGH)

wifi.setmode(wifi.STATION)
--wifi.sta.config("DIGISOL","digi@12345")
wifi.sta.config("JioDongle2_5088AD","mgm9t6qv5j")

wifi.sta.setip({ip="192.168.15.71",netmask="255.255.255.0",gateway="192.168.15.1"})
print("Connected to IP "..wifi.sta.getip())

srv=net.createServer(net.TCP)
srv:listen(80,function(conn)
    conn:on("receive", function(client,request)
        local buf = "";
        local _, _, method, path, vars = string.find(request, "([A-Z]+) (.+)?(.+) HTTP");
        if(method == nil)then
            _, _, method, path = string.find(request, "([A-Z]+) (.+) HTTP");
        end
        local _GET = {}
        if (vars ~= nil)then
            for k, v in string.gmatch(vars, "(%w+)=(%w+)&*") do
                _GET[k] = v
            end
        end
        buf = buf.."<body bgcolor=\"beige\"><h1><font color = \"blue\">Light Pattern Control System</font></h1></body>";
        buf = buf.."<h2><font color = \"DARKSLATEGRAY\">University of Jammu</font></h2>";

        buf = buf.."<font color = \"ROSYBROWN\">* Connected to ESP on "..wifi.sta.getip();
        --buf = buf.."<img src=\"https://www.dropbox.com/pri/get/Public/01058384000.jpg?_subject_uid=76159750&w=AAAMcS12noX

        buf = buf.."</font><br><p><a href=\"?pin=INBLT\"><button>ESP LED</button></a>&nbsp;&nbsp;&nbsp;&nbsp;<a href=\"?pin
        buf = buf.."<p><a href=\"?pin=ALLON\"><button>LIGHTS ON</button></a>&nbsp;&nbsp;&nbsp;&nbsp;<a href=\"?pin=ALLOFF\"
        buf = buf.."<p><a href=\"?pin=TEST\"><button>GO</button></a>&nbsp;&nbsp;&nbsp;&nbsp;<a href=\"?pin=ALLOFF\"><button
```

D:\MTECH 2015 UOJ\DISSERTATION\final\init.lua

```
> w([==[                gpio.write(pin_inblt,gpio.HIGH)]==]);
> w([==[                gpio.write(pin_bulb,gpio.HIGH)]==]);
> w([==[               ]==]);
> w([==[          elseif(_GET.pin == "BULB")then]==]);
> w([==[                gpio.write(pin_led,gpio.LOW)]==]);
> w([==[                gpio.write(pin_inblt,gpio.HIGH)]==]);
> w([==[                gpio.write(pin_bulb,gpio.LOW)]==]);
> w([==[]==]);
> w([==[          elseif(_GET.pin == "ALLON")then]==]);
> w([==[                gpio.write(pin_led,gpio.HIGH)]==]);
> w([==[                gpio.write(pin_inblt,gpio.LOW)]==]);
> w([==[                gpio.write(pin_bulb,gpio.LOW)]==]);
> w([==[]==]);
> w([==[          elseif(_GET.pin == "ALLOFF")then]==]);
> w([==[                gpio.write(pin_led,gpio.LOW)]==]);
> w([==[                gpio.write(pin_inblt,gpio.HIGH)]==]);
> w([==[                gpio.write(pin_bulb,gpio.HIGH)]==]);
> w([==[]==]);
> w([==[          elseif(_GET.pin == "TEST")then]==]);
> w([==[                gui.OpenURL("http://www.google.com")]==]);
> w([==[                ]==]);
> w([==[                           ]==]);
> w([==[          end]==]);
> w([==[          client:send(buf);]==]);
> w([==[          client:close();]==]);
> w([==[          collectgarbage();]==]);
> w([==[      end)]==]);
> w([==[end)]==]);
> file.close();
> dofile("init.lua");
Connected to IP 192.168.15.71
```

2. Select same COM port in the right-hand section, select baud rage as 9600 and hit Open. You will get connection message on the right console.

3. Get respective codes from the Source Code directory (github or CD). Flash init.lua and ldr.lua to the device by browsing the code and then hitting Save&Run in the left section.

## 1.4 Changing parameters and values in LUA code

1. Define SSID and password of the AP in

   init.lua: wifi.sta.config("<ssid>","<password>");

2. Define static IP parameters in

   init.lua: wifi.sta.setip({ip="<ip>",netmask="<subnet mask>",gateway="<gateway>"});

3. Define maximum darkness in the range [0,1023] in

   ldr.lua: dim_max = <your tested value of max darkness>

   This maximum value can be tested by covering the LDR and measuring the value (after circuit part is complete).

4. Change frequency of LDR sensing in

   ```
   ldr.lua: if not tmr.alarm(0,<interval in ms>,1, function()
   ```

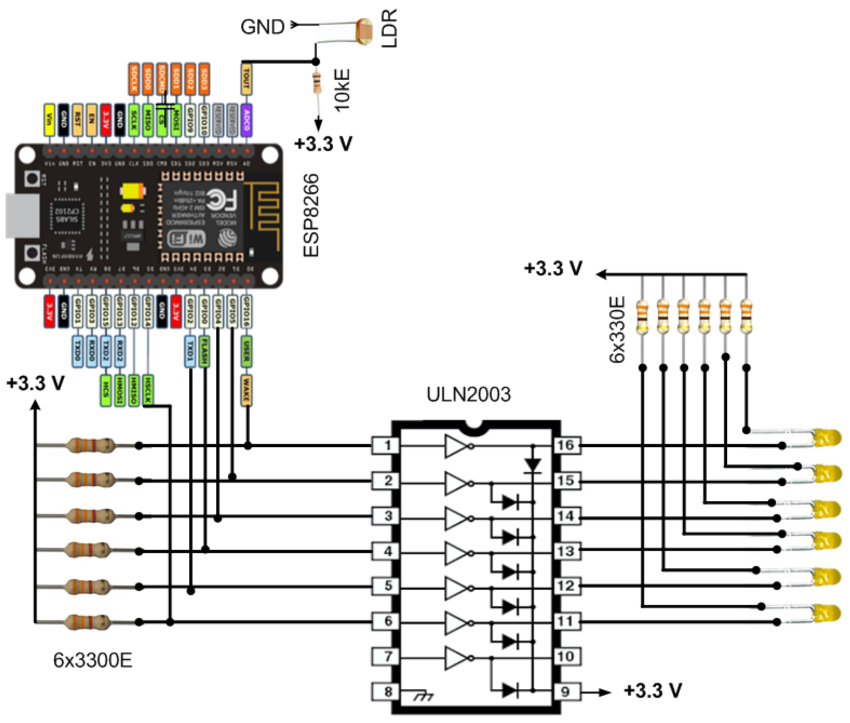
5. The current LDR readings can be read on the above console by connecting the LDR to PC while in operation. The HTTP GET commands described later could also be used to get the value as response.

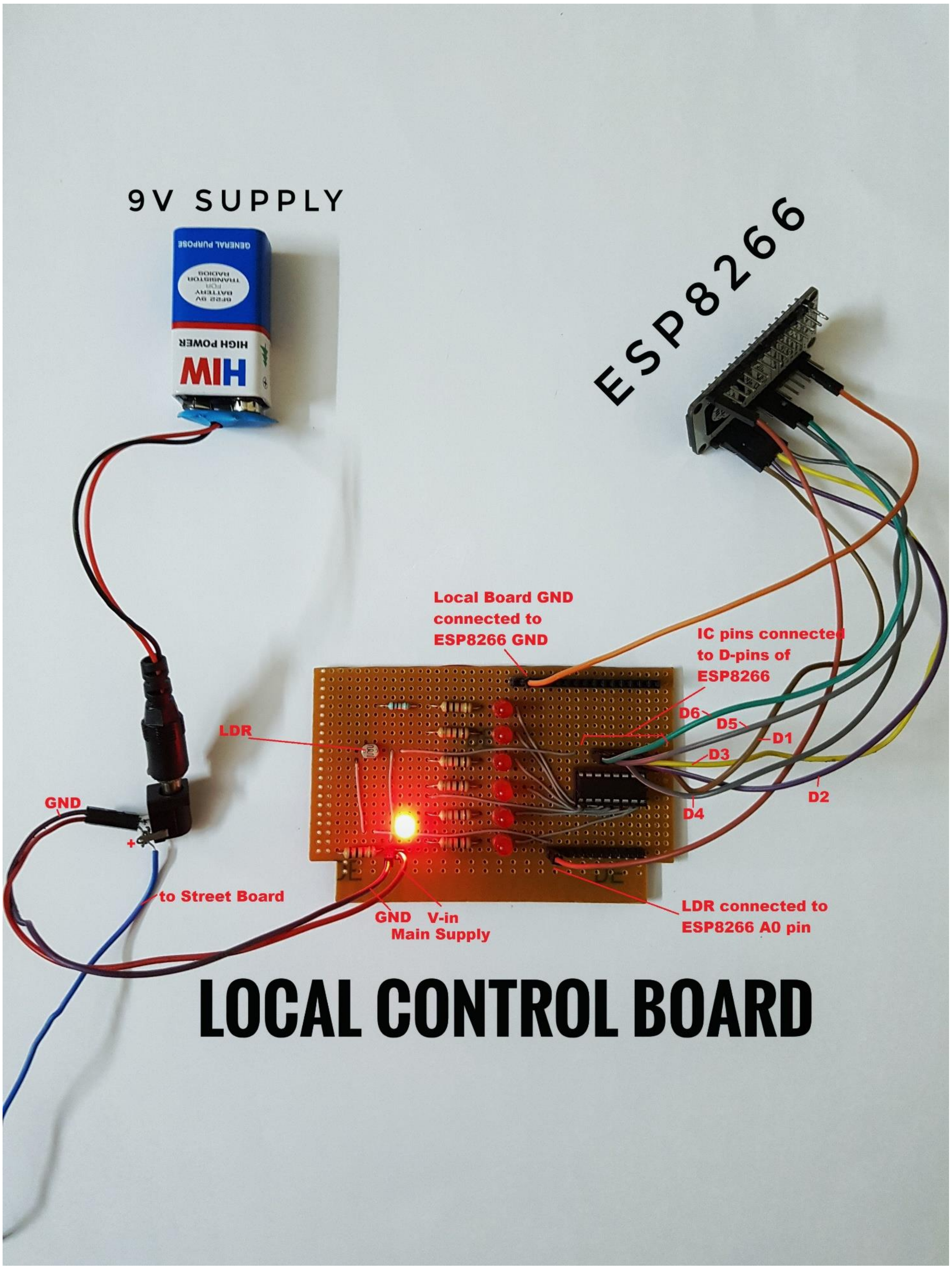
6. You can further change and play with HTTP response messages, lamp numbering and threshold values of LDR reading.

# CHAPTER 2: HARDWARE CIRCUIT

## 2.1 Local Control Board

1. Connect $V_{in}$ and GND with a preferably 9V supply.
2. Connect pins to the left side of the IC to ESP8266 pins numbered from D1 to D6.
3. Make GND common with ESP8266 GND.
4. Connect A0 pin of ESP to LDR socket on the board.
5. Please refer to the Circuit Diagram for initial connections on the board. The +3.3V is replaced by a 9V main supply in order to drive the Street Board.




9V SUPPLY
ESP8266
Local Board GND connected to ESP8266 GND
IC pins connected to D-pins of ESP8266
D6
D5
D1
D3
D2
D4
LDR
GND
+
to Street Board
GND V-in Main Supply
LDR connected to ESP8266 A0 pin
LOCAL CONTROL BOARD

## 2.2 Street Board

1. Connect street lamps on the board to respective pins on the socket to the right side of IC. This way the D1 pin would map to Lamp #1 on the board.

2. Connect power supply of the board to the main supply.

3. Connect GND (or negative terminal) of the lamps to the common GND connected with the main supply.

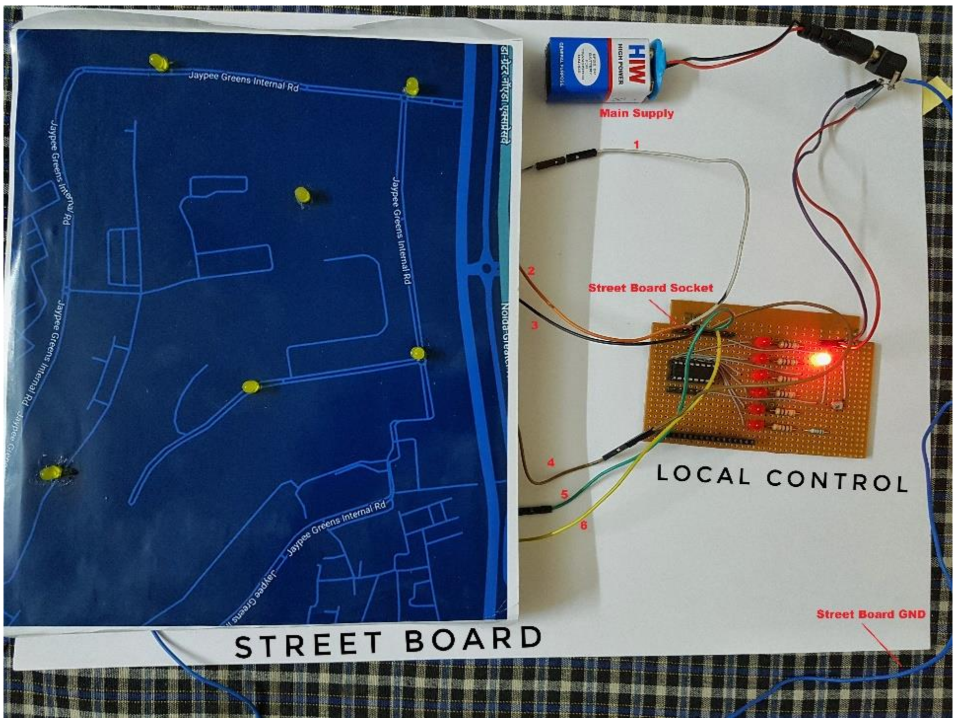
4. Once ESP is powered by the Micro USB port preferably by a Power Bank, you can test the street board lamps by replacing the D pins with the 3V3 pin on the ESP.

5. After testing and fixing the circuit if there was any issue with the connections, replace the 3V3 pin with the respective D pins.

6. Variate light intensity over the LDR and watch lamps adjust overall brightness.

7. For every lamp toggled in the street model board, respective LED indicates this on the local board. This can be used for troubleshooting the connections between the code, mobile application and the street model.

8. **The hardware setup is complete now. You can control and monitor the street system on the mobile application once you have set up the software part. This part completes the automation of the street light by using environmental light intensity.**



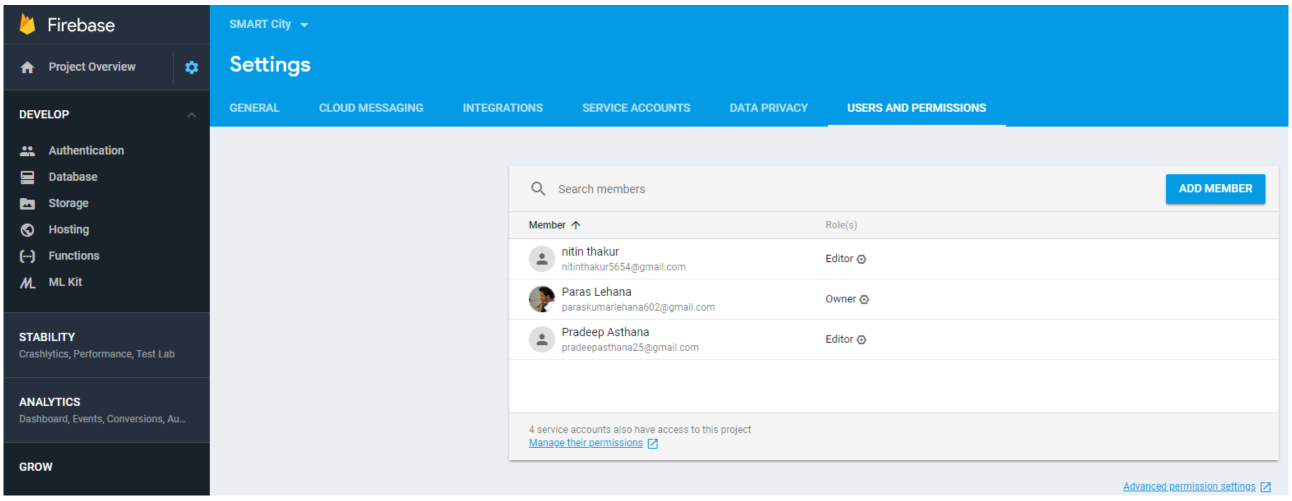
# CHAPTER 3: CLOUD DATABASE

## 3.1 Logging into Firebase & Authentication for application

1. You need to be an admin to add lamps to the database. For only monitoring lamp data or any other specific privileges, request credentials from the admin.

2. For testing credentials, you can request testing credentials from the admins or owners of the project. Your Google email address needs to be registered in the IAM panel of this Firebase Project.

3. This way previous admins can add new admins or users for the street area to be controlled or monitored.

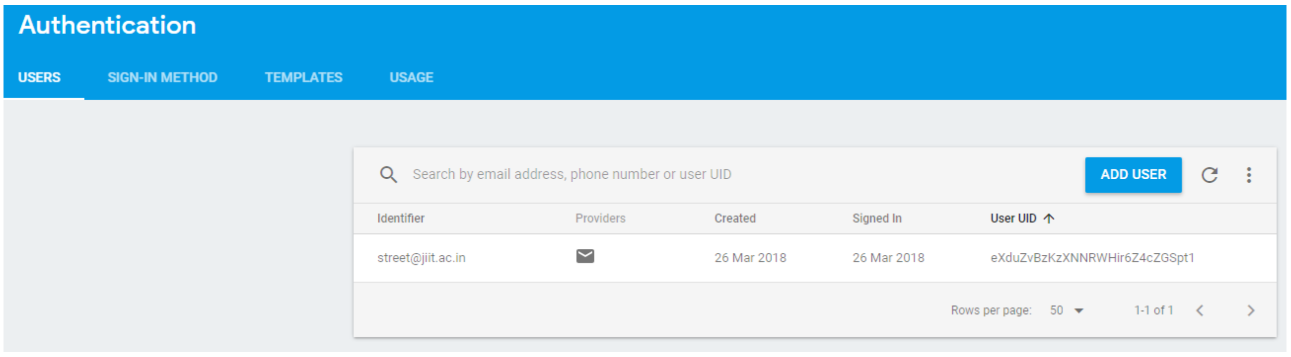


| Member | Role(s) |
|---|---|
| nitin thakur<br>nitinthakur5654@gmail.com | Editor |
| Paras Lehana<br>paraskumarlehana602@gmail.com | Owner |
| Pradeep Asthana<br>pradeepasthana25@gmail.com | Editor |

4 service accounts also have access to this project
Manage their permissions

Advanced permission settings

4. Add users or admins for the mobile application in the Authentication tab.



| Identifier | Providers | Created | Signed In | User UID |
|---|---|---|---|---|
| street@jiit.ac.in | ✉ | 26 Mar 2018 | 26 Mar 2018 | eXduZvBzKzXNNRWHir6Z4cZGSpt1 |

Rows per page: 50 1-1 of 1

## 3.2 Adding Lamps

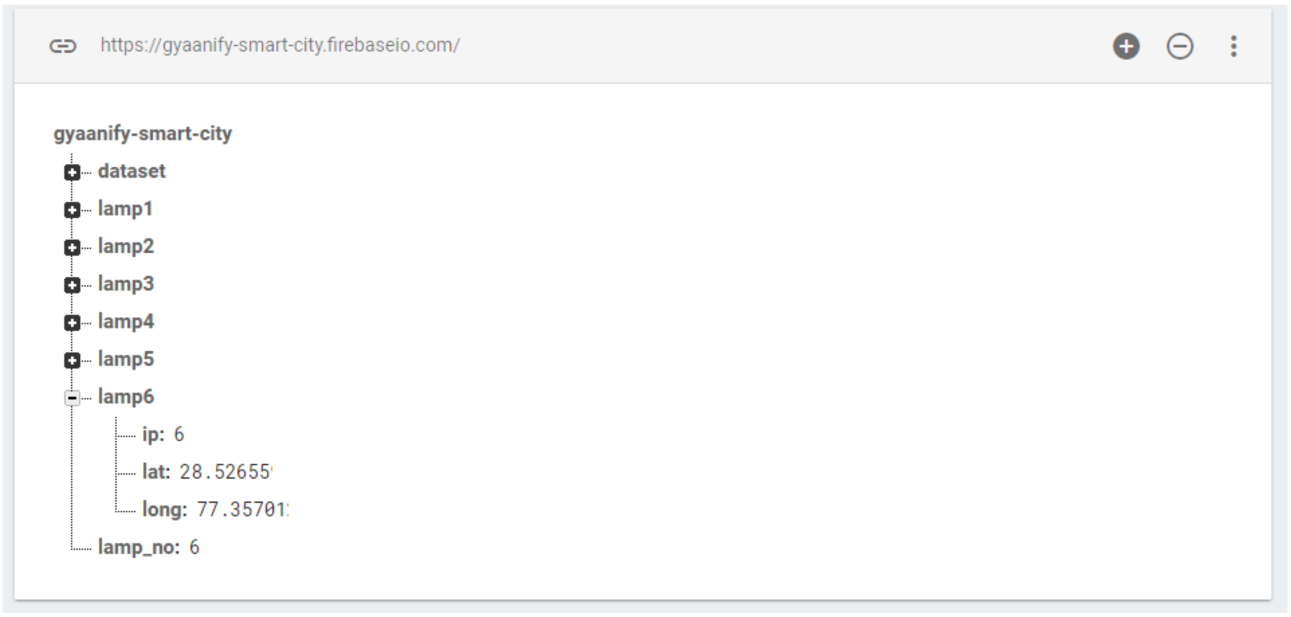
1. Extract coordinates (latitude, longitude) of the location where the lamp is installed from Google Maps. You can also copy the coordinates flashed while clicking on the StreetMap in the ‘Adding Lamps’ part of the application.

2. Click ‘+’ besides gyaanify-smart-city.

3. Enter values for keys below as:
   ip: <lamp no. as per the circuit>
   lat: <latitude of the lamp location>
   long: <longitude of the lamp location>

4. Increment lamp_no for each lamp added. You can add lamps without updating this variable to make the added lamp not to show on the application screen for the time being.

5. As soon as you press Enter and data is stored successfully (indicated by yellow to green transmission of the tuple colour), you can notice the lamp added on the respective location in your application.

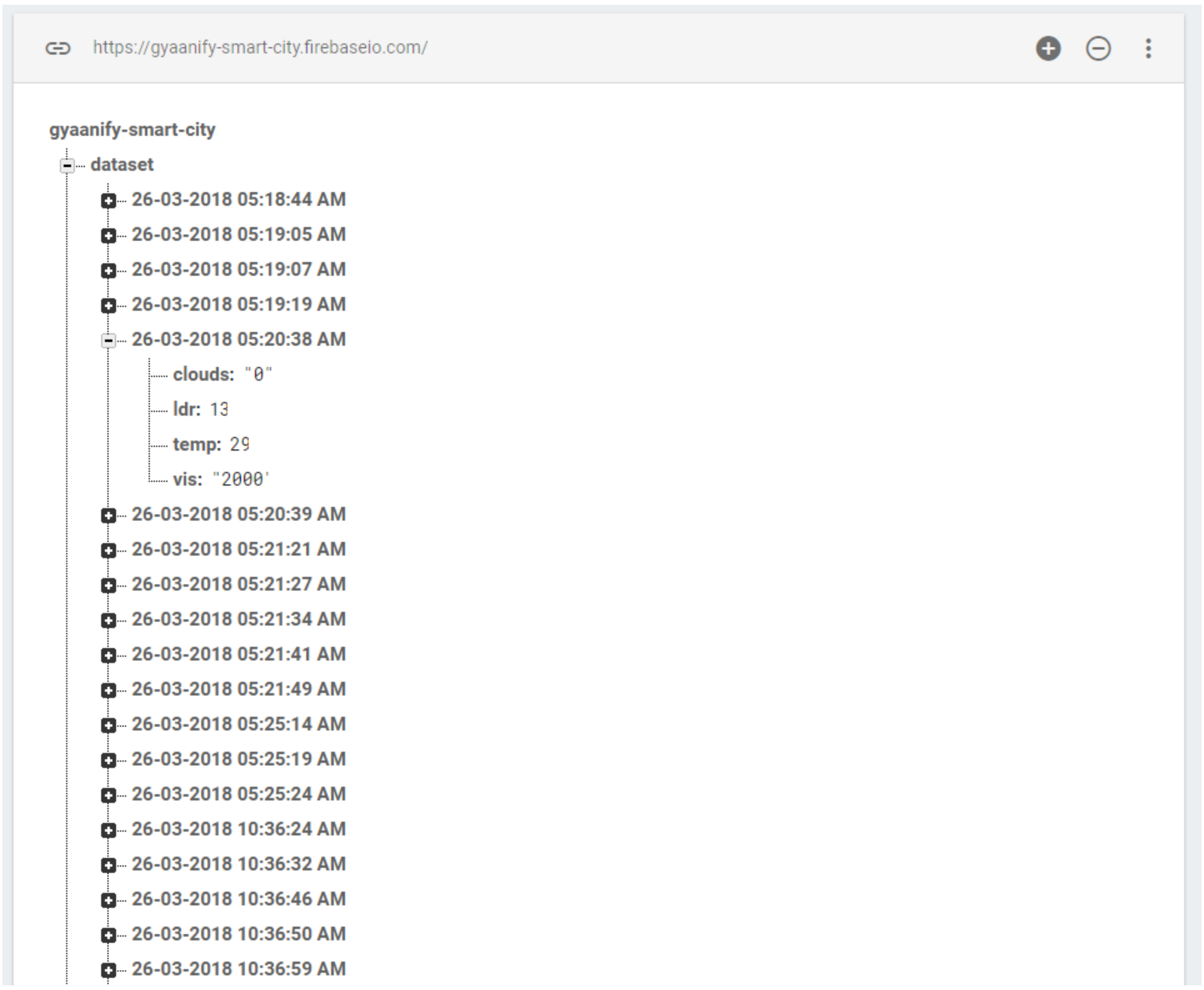


```
https://gyaanify-smart-city.firebaseio.com/

gyaanify-smart-city
  dataset
  lamp1
  lamp2
  lamp3
  lamp4
  lamp5
  lamp6
      ip: 6
      lat: 28.52655
      long: 77.35701
  lamp_no: 6
```

## 3.3 Dataset

1. You can view the collected data in the dataset field. This includes the sensor and Weather API data which is uploaded to the cloud every time Master Control (mobile application) is used to control bulbs. This way the model can further train itself through a supervised learning algorithm.



https://gyaanify-smart-city.firebaseio.com/

```
gyaanify-smart-city
  dataset
    26-03-2018 05:18:44 AM
    26-03-2018 05:19:05 AM
    26-03-2018 05:19:07 AM
    26-03-2018 05:19:19 AM
    26-03-2018 05:20:38 AM
      clouds: "0"
      ldr: 13
      temp: 29
      vis: "2000"
    26-03-2018 05:20:39 AM
    26-03-2018 05:21:21 AM
    26-03-2018 05:21:27 AM
    26-03-2018 05:21:34 AM
    26-03-2018 05:21:41 AM
    26-03-2018 05:21:49 AM
    26-03-2018 05:25:14 AM
    26-03-2018 05:25:19 AM
    26-03-2018 05:25:24 AM
    26-03-2018 10:36:24 AM
    26-03-2018 10:36:32 AM
    26-03-2018 10:36:46 AM
    26-03-2018 10:36:50 AM
    26-03-2018 10:36:59 AM
```

2. The dataset or any other fields can be exported to a JSON or spreadsheet format by selecting the options from the three-dotted-menu.

# CHAPTER 4: MOBILE APPLICATION & REMOTE ACCESS

## 4.1 Logging into the App

1. With the credentials discussed in 3.1.4, login into the app. The application will remember the last login state for subsequent sessions now.

2. For trial, use these credentials:

   Email: street@jiit.ac.in
   Password: try@123

3. Prior to the logging, you should have changed the gateway part and IP part in the Android code in StreetMap.java.

## 4.2 Controlling Lamps

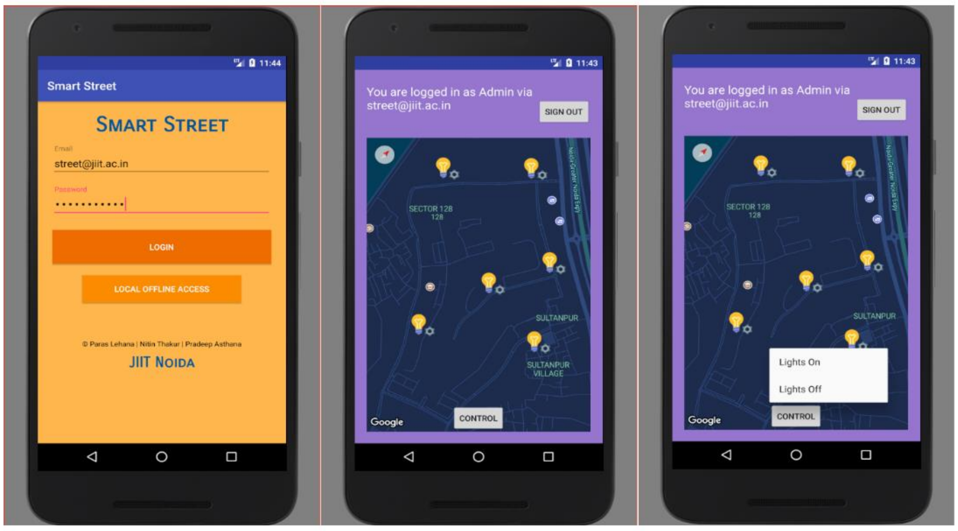
1. Tap the lamp you want to control on the StreetMap. The IP address on the toast will denote the lamp no. corresponding to the data stored on Cloud.

2. Tap CONTROL now and select the desired option – ON/OFF.

3. Watch the lamps in remote access mode on the boards.

4. You'll see a toast about current Temperature, Visibility and Cloudiness of the area. This is fetched from the Weather API. Change coordinates of the location in the API URL in the same file.

5. The weather data along with timestamp and LDR intensity is sent to Cloud for dataset purposes.


Email: street@jiit.ac.in
© Paras Lehana | Nitin Thakur | Pradeep Asthana
You are logged in as Admin via street@jiit.ac.in
You are logged in as Admin via street@jiit.ac.in

# CHAPTER 5: SENSING & DATASET

## 5.1 Programming TelosB

1. Navigate to tinyos directory or the source code folder and search for Oscilloscope folder.

2. Connect the Oscilloscope (Remote device) to the PC.

3. Open Oscilloscope.nc.

4. Change default sensor by:

   Temperature: Add “new SensirionSht11C()” in components and change

   “OscilloscopeC.Read -> Sensor” to
   “OscilloscopeC.Read -> SensirionSht11C().Temperature”

   Total Solar Radiation: Add “HamamatsuS10871TsrC() as PhotoTsr” in components and change

   “OscilloscopeC.Read -> Sensor” to
   “OscilloscopeC.Read -> PhotoTsr”

   Total Solar Radiation: Add “HamamatsuS10871ParC() as PhotoTsr” in components and change

   “OscilloscopeC.Read -> Sensor” to
   “OscilloscopeC.Read -> PhotoPar”

5. Open terminal in the same directory and run “make telosb install”

6. Connect BaseStation (Master node) to the PC

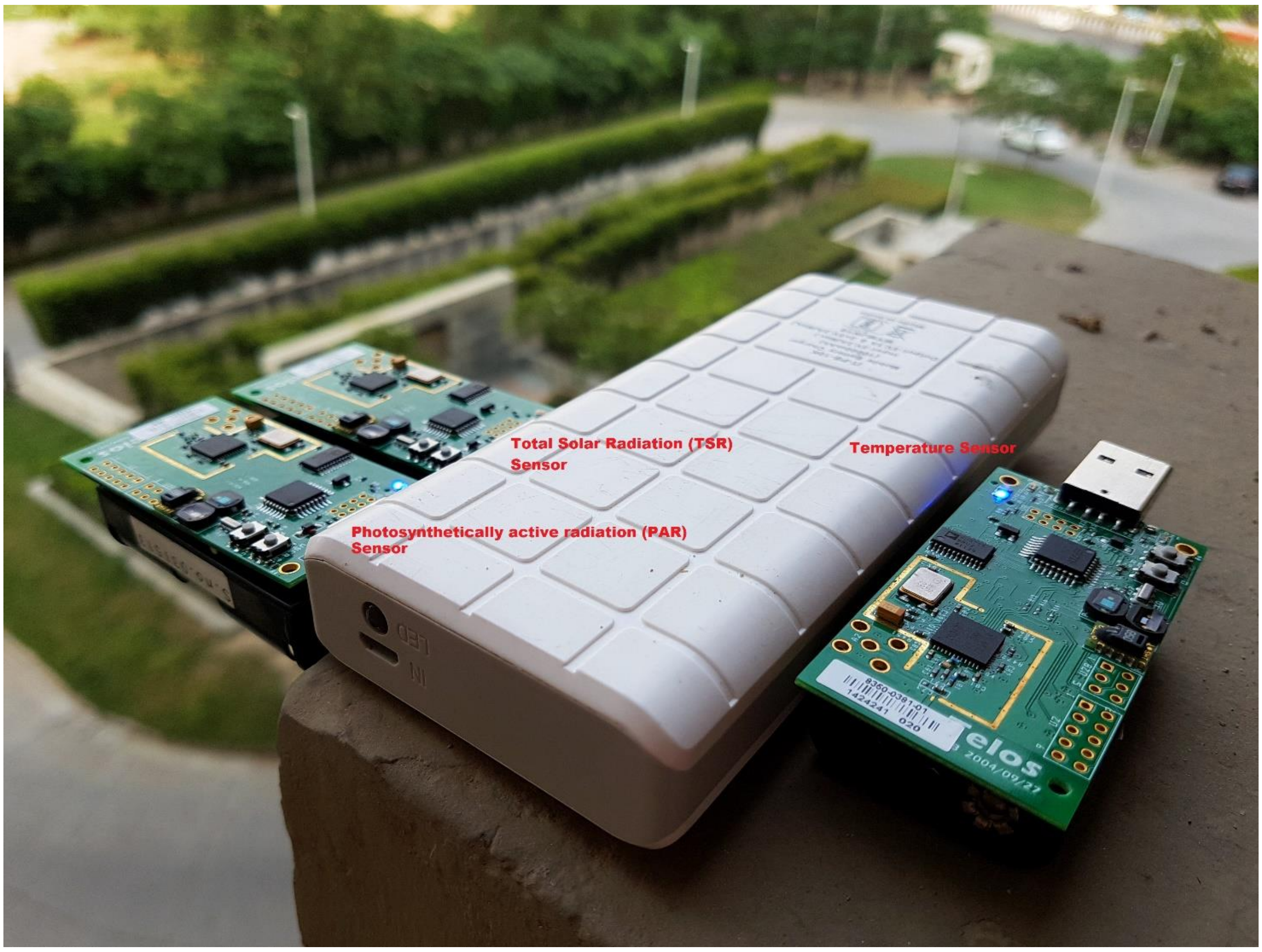
7. Search BaseStation folder now. Run “make telosb install”

8. Deploy the Oscilloscope.

9. Navigate to test/TestSerial and run motelist to get device name. Note the USB port number.

10. Run "java Tinyos.sf.SerialForwarder -comm serial@/[device name]:telosb >> example.txt" will output file "example.txt" with packets where device name is the USB port number noted above.

11. Omit the redirection operator to see output in the terminal. Press CRTL+Z anytime to stop receiving packets.


Total Solar Radiation (TSR) Sensor
Temperature Sensor
Photosynthetically active radiation (PAR) Sensor

## 5.2 Processing Packets

1. Navigate to the scripts folder.

2. Run “g++ script.cpp”

3. Run "./a.out < inputfile.txt > outputfile.txt" output a processed file "outputfile.txt".

4. Use scripts accordingly to the type of data processing needed. Different types of processed data are imported in CSV format in the directory.

5. This dataset will be further used to chart analysis and learning.