IoT based Smart Street Light Controlling System

Prince Prag
180102051
Electronics and Communication
Engineering
Indian Institute of Technology Guwahati
Assam
Email: prag@iitg.ac.in

***Abstract*—The 21st century is experiencing a large upsurge in the use of Internet of Things (IoT) technologies and smart devices.With the increasing availability of these devices, the concept of a smart city is taking shape. smart streetlighting is one of the core components of a smart city. As there is a large distribution of streetlights all accross the city, establishing a network between them will provide a great potential to serve as a readily available infrastructure supporting additional sensing and control devices, such as environmental sensors, traffic sensors and/or IP cameras, to provide city-wide services. This paper focuses on the basis model of smart street light controlling system and the different communication technologies that can be used to communicate between the server and clients.**

***Index Terms*—IoT, Smart streetlighting, TCP/IP Protcol, Rx-Tx Serial Communication, Google Firebase, Android application**

I. Main Objective

The main objective of this project is automating the control of streetlights and failure detection in it. The streetlights will switch their states with respect to any impending motion in the neighbourhood and the intensity of natural light available. This will prevent unnecessary wastage of electricity and will also reduce human dependency to control the streetlights. The authority can monitor the status of the streetlights and the motion in the streets using a webpage or an Android application.

II. Implemented attributes

A. Server-Client Communication using TCP/IP protocol

The client1 communicates with server using TCP/IP protocol. It sends the PIR reading, LDR reading, Bulb reading and Bulb State to the server by making a HTTP request and the readings are appended in the string in JSON format.

B. Server-Client Communication using Rx-Tx Serial Communication

The client2 communicates with server using TCP/IP protocol. It sends the PIR reading, LDR reading, Bulb reading and Bulb State to the server by sending the readings in the form a string using the serial port which is the parsed in the server to get the appropriate readings.

C. Server-Google Firebase Data transfer

The server pushes the realtime readings received from both the clients to google firebase which is used in the dashboard in the Android application. Besides it also pushes the data of regular periodic checks by both the clients so that the authority can monitor the status of the streetlights.

D. Text Message

The server with the help of the twillio api sends a text message to the authority stating the status of streetlights during the periodic checks.

E. Android Application

An android application is developed that fetches the data from firebase and helps the authority to monitor the motion around streetlights and the status of streetlights at different periodic intervals.

III. Configuration Diagram

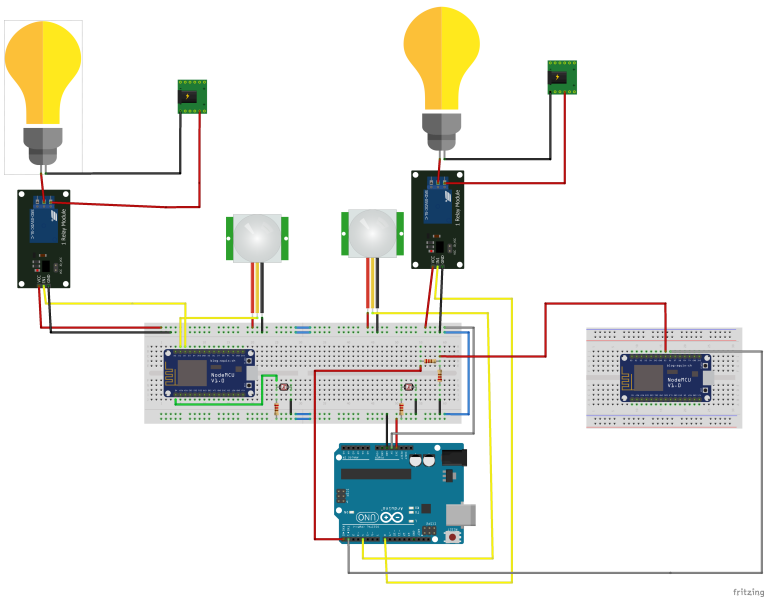

Fig. 1: The configuration diagram of the circuit

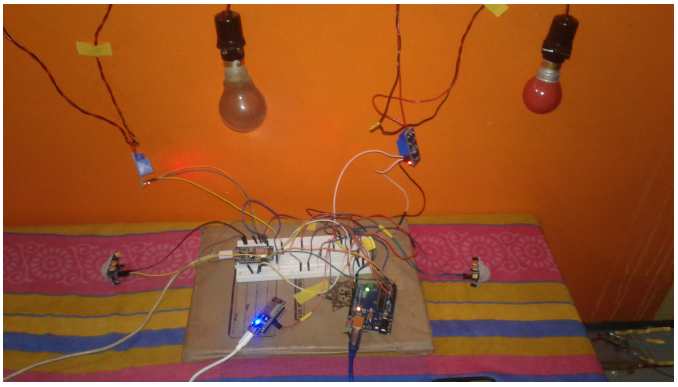

Fig. 2: The Project Setup

IV. Sample Outputs

These are the following sample outputs in the serial monitor and the real project:

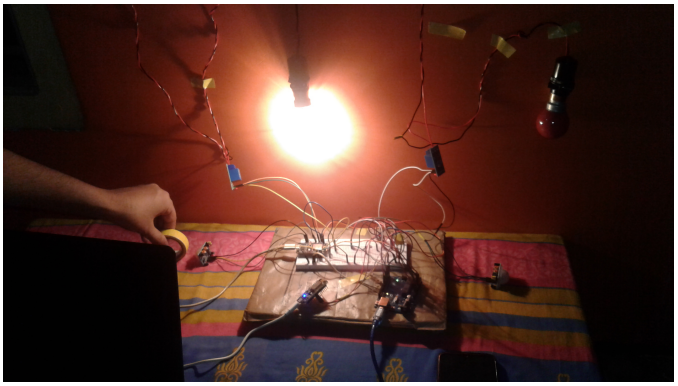

Fig. 3: Client 1 Motion

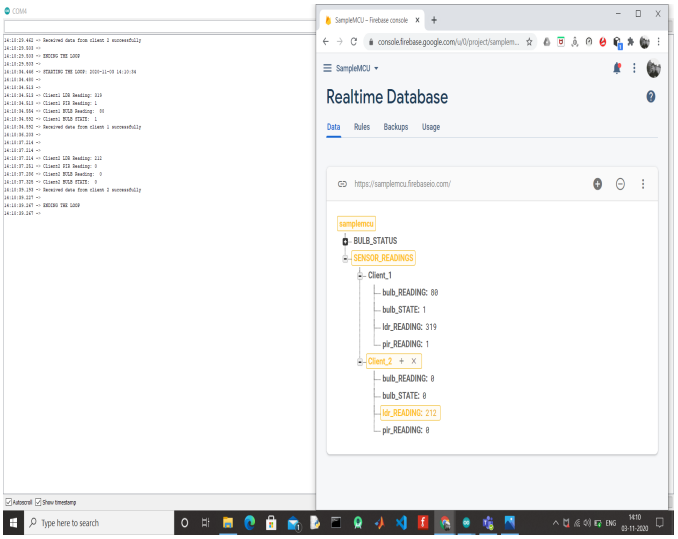

Fig. 4: Client1 Motion Readings

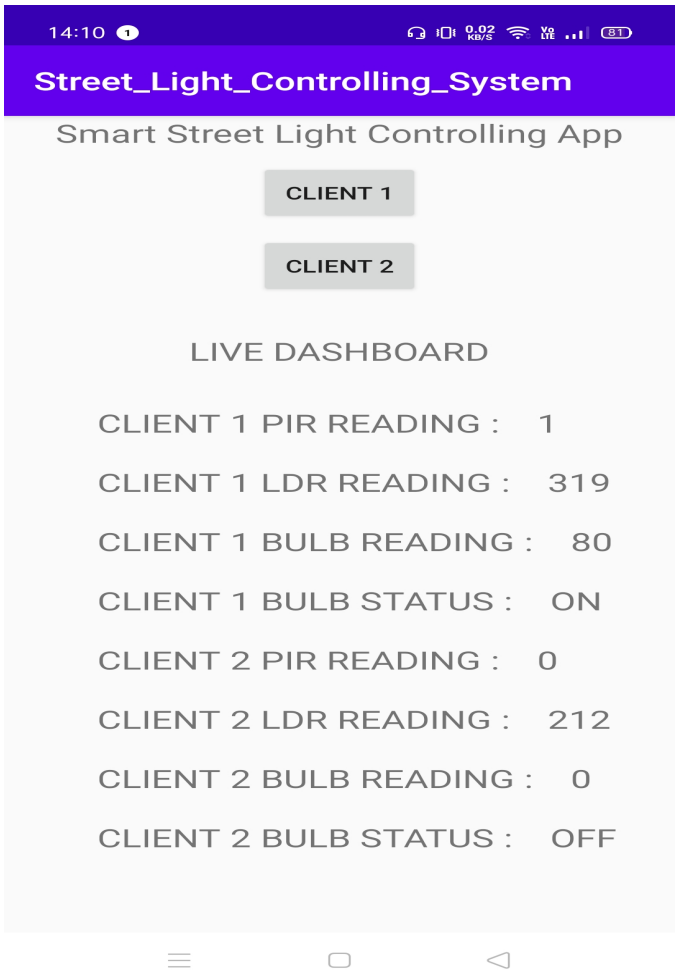

Fig. 5: Client1 Motion Readings on App

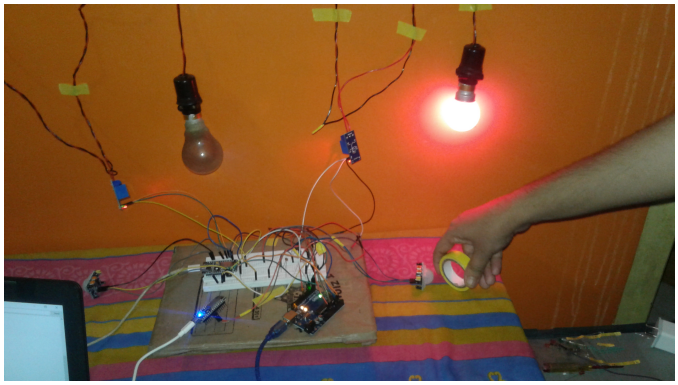

Fig. 6: Client2 Motion

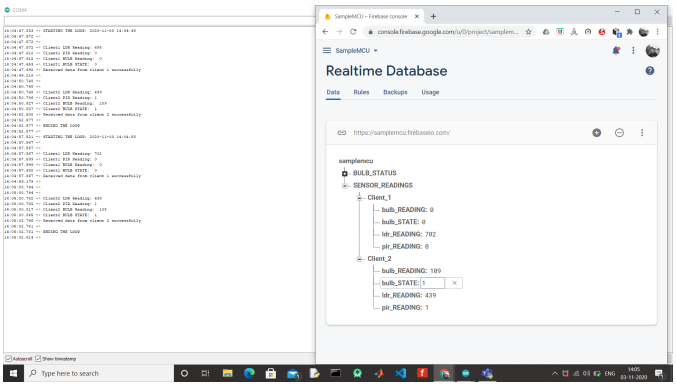

Fig. 7: Client2 Motion Readings

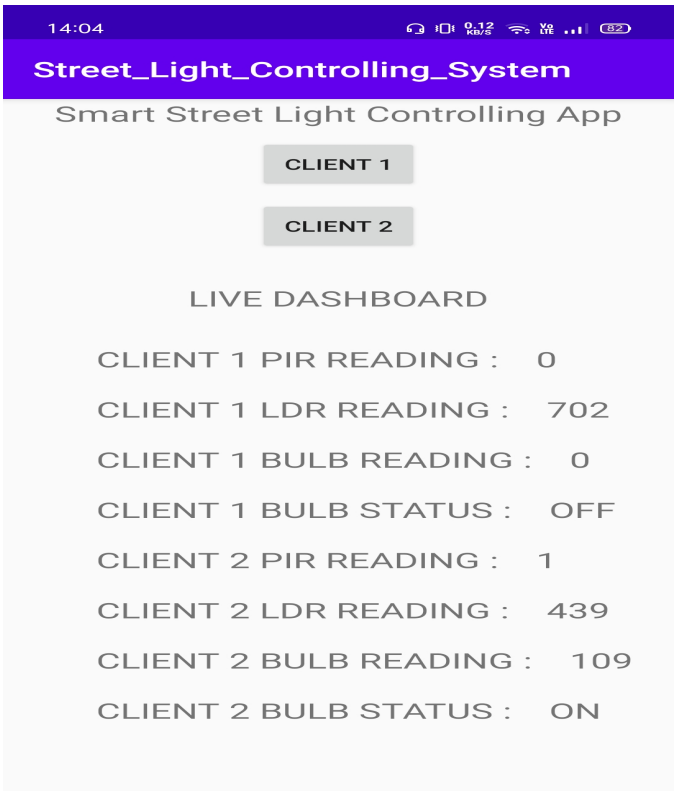

Fig. 8: Client2 Motion Readings on Android app

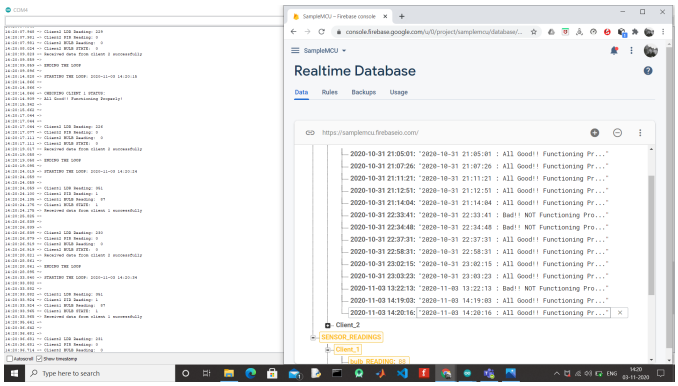

Fig. 9: Client1 Status Readings on Serial Monitor

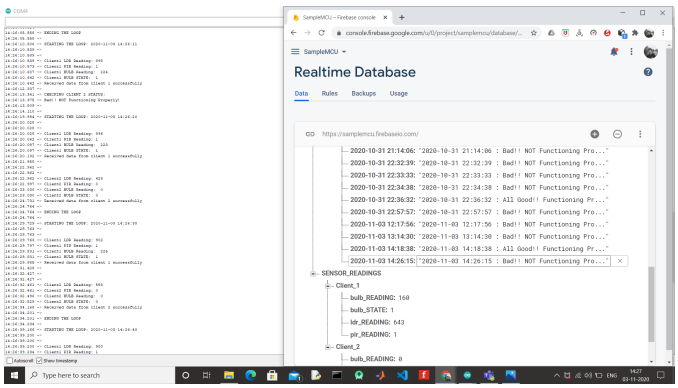

Fig. 10: Client2 Status Readings on Serial Monitor

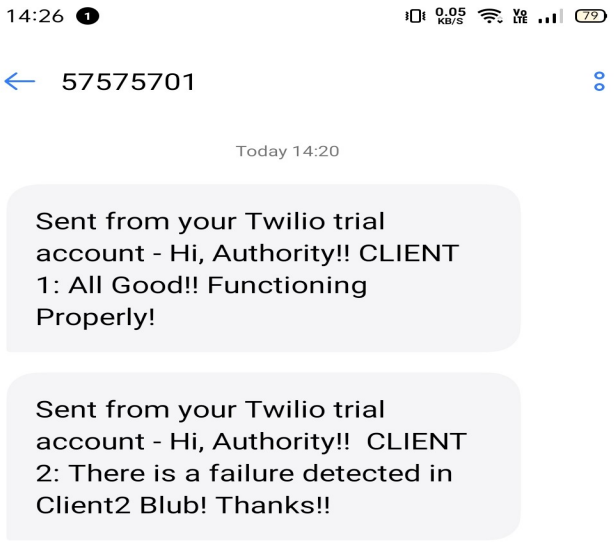

Fig. 11: Text Message on Phone

. V. CODES

A. Server

Following is the code to be uploaded in the server:

```
#include < FirebaseArduino.h >
#include < ESP8266WiFi.h >
#include < ESP8266WebServer.h >
#include < ArduinoJson.h >
#include < SoftwareSerial.h >
#include < ESPDateTime.h >
#define FIREBASE_HOST ”samplemcu.firebaseio.com”
#define FIREBASE_AUTH ”CTBXcH0fAhCL1LvRu3gL0V
XAEpgulblo7L7Xfq9Y”
char serverr[] = ”api.thingspeak.com”;
#define D6 12
#define D5 14
#define led0 16 //D0
#define led1 5 //D1
#define led2 4 //D2
#define led3 0 //D3
int client1_LDR_READING=0;
int client1_PIR_READING=0;
int client1_BULB_READING=0;
int client1_BULB_STATE=0;
int client1_BULB_STATUS;
int client2_LDR_READING=0;
int client2_PIR_READING=0;
int client2_BULB_READING=0;
int client2_BULB_STATE=0;
int client2_BULB_STATUS;
DynamicJsonBuffer jsonBuffer,jjsonBuffer;
SoftwareSerial s(D6, D5);
const char ssid = ”NODEMCU_WIFI”;
const char password = ”NODEMCU_PASSWORD”;
const String apiKey = ”RWN78RXG18O6OBM2”;
const String sendNumber = ”+919661268236”;
const char* host = ”maker.ifttt.com”;
const int httpsPort = 443;
String sensor_values,bulbs_values;
ESP8266WebServer server(80);

void setupDateTime()
{
DateTime.setTimeZone(5);
DateTime.begin();
if (!DateTime.isTimeValid())
{
Serial.println(”Failed to get time from server.”);
return;
}
}

void handleSentVarClient1()
{
if (server.hasArg(”sensor_reading”))
{
sensor_values = server.arg(”sensor_reading”);
}
JsonObject& root = jsonBuffer.parseObject(sensor_values);
String senttime;
if(root.success())
{
client1_LDR_READING= root[”LDR_READING”].as<int>();
client1_PIR_READING= root[”PIR_READING”].as<int>();
client1_BULB_READING= root[”BULB_READING”].as<int>();
client1_BULB_STATE= root[”BULB_STATE”].as<int>();
Serial.println(” ”);
Serial.print(”Client1 LDR Reading: ”);
Serial.println(client1_LDR_READING);
Serial.print(”Client1 PIR Reading: ”);
Serial.println(client1_PIR_READING);
Serial.print(”Client1 BULB Reading: ”);
Serial.println(client1_BULB_READING);
Serial.print(”Client1 BULB STATE: ”);
Serial.println(client1_BULB_STATE);
Serial.println(”Received data from client 1 successfully”);
Firebase.setInt(”/SENSOR_READINGS/Client_1/LDR_READING”,
client1_LDR_READING);
Firebase.setInt(”/SENSOR_READINGS/Client_1/pir_READING”,
client1_PIR_READING);
Firebase.setInt(”/SENSOR_READINGS/Client_1/bulb_READING”,
client1_BULB_READING);
Firebase.setInt(”/SENSOR_READINGS/Client_1/bulb_STATE”,
client1_BULB_STATE);
Serial.println(” ”);
}
else
{
Serial.println(” ”);
Serial.println(”Failure in getting data from client 1”);
Serial.println(” ”);
}
}

void handleClient1Check()
{
Serial.println(” ”);
Serial.println(”CHECKING CLIENT 1 STATUS:”);
if (server.hasArg(”bulb_status”))
{
bulbs_values = server.arg(”bulb_status”);
}
JsonObject& root2 = jsonBuffer.parseObject(bulbs_values);
String p;
if(root2.success())
{
client1_BULB_STATUS= root2[”CLIENT1_BULB_STATUS”].
as< int > ();
}
if(client1_BULB_STATUS==0)
{
```

```
p=”Bad!! NOT Functioning Properly!”;
Serial.println(”Bad!! NOT Functioning Properly!”);
sendSMS(sendNumber, URLEncode(”Hi,Authority!!
CLIENT 1: There is a failure
detected in Client1 Blub! Thanks!!”));
}
else
{
p=”All Good!! Functioning Properly!”;
Serial.println(”All Good!! Functioning Properly!”);
sendSMS(sendNumber, URLEncode(”Hi, Authority!!
CLIENT 1: All Good!!
Functioning Properly!”));
}
Serial.println(” ”);
Firebase.setString(”/BULB_STATUS/Client_1/”+DateTime.
toString(),DateTime.toString()+” : ”+p);
}


void sendSMS(String number,String message) {
WiFiClient client;
if (client.connect(serverr, 80))
{
String urrl=”/apps/thinghttp/send_request?api_key=”;
urrl+=apiKey;
urrl+=”&number=”;
urrl+=number;
urrl+=”&message=”;
urrl+=message;
client.print(String(”GET ”) + urrl + ” HTTP/1.1\r\n” +
”Host:” + serverr + ”\r\n” + ”Connection: close\r\n\r\n”);
}
else
{
Serial.println(F(”Connection failed”));
}
int p = millis()/1000;
while (client.connected())
{
if ( client.available())
{
char c = client.read();
Serial.print(c);
}
else break;
if(millis()- p * 1000 > 5000) break;
}
Serial.println();
client.stop();
}

String URLEncode(const char* msg)
{
const char *hex = ”0123456789abcdef”;
String encodedMsg = ””;
while (*msg!=’\0’)
{
if( (’a’ <= *msg && *msg <= ’z’)
|| (’A’ <= *msg && *msg <= ’Z’)
|| (’0’ <= *msg && *msg <= ’9’))
{
encodedMsg += *msg;
}
else
{
encodedMsg += ’%’;
encodedMsg += hex[*msg >> 4];
encodedMsg += hex[*msg & 15];
}
msg++;
}
return encodedMsg;
}

void setup()
{
Serial.begin(9600);
s.begin(9600);
WiFi.mode(WIFI_AP_STA);
WiFi.softAP(ssid,password);
WiFi.begin(”PRINCE PRAG”,”@Piyush2005”);
IPAddress myIP = WiFi.softAPIP();
Serial.println(WiFi.localIP());
Serial.println(WiFi.softAPIP());
Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
pinMode(led0, OUTPUT);
pinMode(led1, OUTPUT);
server.on(”/data/”, HTTP_GET, handleSentVarClient1);
server.on(”/checking/”, HTTP_GET, handleClient1Check);
server.begin();
}

void loop()
{
setupDateTime();
Serial.println(”STARTING THE LOOP:”
+DateTime.toString());
Serial.println(” ”);
server.handleClient();
String msg = s.readString();
if(msg[0]==’*’)
{
Serial.println(”CHECKING CLIENT 2 STATUS:”);
int lt= msg.length();
client2_BULB_STATUS = msg.substring(1,lt-1).toInt();
String p;
if(client2_BULB_STATUS==0)
{
p=”Bad!! NOT Functioning Properly!”;
Serial.println(”Bad!! NOT Functioning Properly!”);
sendSMS(sendNumber,URLEncode(”Hi,Authority!! CLIENT
```

```
2: There is a failure detected in Client2 Blub! Thanks!!"));
}
else
{
p="All Good!! Functioning Properly!";
Serial.println("All Good!! Functioning Properly!");
sendSMS(sendNumber,URLEncode("Hi,Authority!! CLIENT
2: All Good!! Functioning Properly!"));
}
Serial.println(" ");
Firebase.setString("/BULB_STATUS/Client_2/"+
DateTime.toString(),DateTime.toString()+" : "+p);
}
else
{
int lth=msg.length();
String key="",value="",s1="";
bool k=true,v=false;
Serial.println(" ");
for(int i=0;i<lth;i++)
{
if(msg[i]=='?'&& s1.length()>0 && k)
{
key=s1;
k=false;
v=true;
s1="";
}
else if(msg[i]=='?'&& s1.length()>0 && v)
{
value=s1;
k=true;
v=false;
s1="";
if(key=="LDR_READING")
client2_LDR_READING=value.toInt();
else if(key=="PIR_READING")
client2_PIR_READING=value.toInt();
else if(key=="BULB_READING")
client2_BULB_READING=value.toInt();
else if(key=="BULB_STATE")
client2_BULB_STATE=value.toInt();
}
else if(k)
{
s1+=msg[i];
}
else if(v)
{
s1+=msg[i];
}
}
Serial.println(" ");
Serial.print("Client2 LDR Reading: ");
Serial.println(client2_LDR_READING);
Serial.print("Client2 PIR Reading: ");
Serial.println(client2_PIR_READING);
Serial.print("Client2 BULB Reading: ");
Serial.println(client2_BULB_READING);
Serial.print("Client2 BULB STATE: ");
Serial.println(client2_BULB_STATE);
Firebase.setInt("/SENSOR_READINGS/Client_2/ldr_READING",
client2_LDR_READING);
Firebase.setInt("/SENSOR_READINGS/Client_2/pir_READING",
client2_PIR_READING);
Firebase.setInt("/SENSOR_READINGS/Client_2/bulb_READING",
client2_BULB_READING);
Firebase.setInt("/SENSOR_READINGS/Client_2/bulb_STATE",
client2_BULB_STATE);
Serial.println("Received data from client 2 successfully");
Serial.println(" ");
Serial.println("ENDING THE LOOP");
Serial.println(" ");
}
delay(5000);
}
```

B. Client1_NodeMCU

Following is the code to be uploaded in the Client_1(NodeMCU):

```
#include <ESP8266WiFi.h>
#include <ESPDateTime.h>
String str;
int PIR=5; //d1
int LDR=A0;
int PIR_VALUE=0;
int LDR_VALUE=0;
int threshold_value=300;
int BULB=4;
int BULB_VALUE=0;
int BULB_STATE=0;
long previous=0;
int threshold=400;
const char *ssid= "NODEMCU_WIFI";
const char *password= "NODEMCU_PASSWORD";
int sensorValue0 = 0;
int sensorValue1 = 0;
int day_threshold=300;

void setup()
{
Serial.begin(115200);
delay(10);
pinMode(PIR,INPUT);
pinMode(LDR,INPUT);
pinMode(BULB,OUTPUT);
// This is a client NodeMCU that connects to
the server NodeMCU
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
```

```
while (WiFi.status() != WL_CONNECTED) {
delay(500);
}
}

void loop()
{
Serial.println("Starting of loop");
WiFiClient client;
const char * host = "192.168.4.1";
//default IP address of the server NodeMCU
const int httpPort = 80;
if (!client.connect(host, httpPort))
{
Serial.println("connection failed");
delay(2000);
return;
}
if(millis()- previous*1000 > 3600000)
{
Serial.println("Performing C1 check");
delay(10000);
previous = millis()/1000;
digitalWrite(BULB,HIGH);
delay(20000);
LDR_VALUE=analogRead(LDR);
String urrl = "/checking/";
urrl + ="?bulb_status=";
urrl + ="{C̈LIENT1_BULB_STATUS:̈sensor0_value}";
if(LDR_VALUE > threshold)
{
Serial.println("Client 1 Bulb not functioning properly");
urrl.replace("sensor0_value","0");
}
else
{
Serial.println("Client 1 Bulb functioning properly");
urrl.replace("sensor0_value","1");
}
client.print(String("GET ") + urrl + " HTTP/1.1\r\n" +
"Host:" + host + "\r\n" + "Connection: close\r\n\r\n");
}
else
{
LDR_VALUE=analogRead(LDR);
PIR_VALUE=digitalRead(PIR);
if(PIR_VALUE==HIGH&&LDR_VALUE>day_threshold)
{
BULB_VALUE= map(LDR_VALUE,0,1023,0,255);
analogWrite(BULB,BULB_VALUE);
}
else
{
digitalWrite(BULB,LOW);
}
BULB_STATE=BULB_VALUE==0?0:1;
String url = "/data/";
url += "?sensor_reading=";
url += "{\"LDR_READING\":\"sensor0_value\",
\"PIR_READING\":\"sensor1_value\",
\"BULB_READING\":\"output_value\",\"BULB_STATE\"
: \"output_state\"}";
url.replace("sensor0_value", String(LDR_VALUE));
url.replace("sensor1_value", String(PIR_VALUE));
url.replace("output_value", String(BULB_VALUE));
url.replace("output_state", String(BULB_STATE));
// This will send the request to the server to get the specific
information
client.print(String("GET ") + url + " HTTP/1.1\r\n" +
"Host:" + host + "\r\n" + "Connection: close\r\n\r\n");
Serial.println("Sending following data to server");
Serial.print("Client1 LDR Reading: ");
Serial.println(LDR_VALUE);
Serial.print("Client1 PIR Reading: ");
Serial.println(PIR_VALUE);
Serial.print("Client1 BULB Reading: ");
Serial.println(BULB_VALUE);
Serial.print("Client1 BULB STATE: ");
Serial.println(BULB_STATE);
Serial.println("Response Received from server: ");
while (client.connected() && client.available())
{
String line = client.readStringUntil('\n');
Serial.println(line);
delay(1000);
}
delay(2000);
client.stop();
Serial.println("");
}
}
```

C. *Client2_Arduino*

Following is the code to be uploaded in the Client_2(Arduino UNO):

```
#include <ArduinoJson.h>
String str;
int PIR=4;
int LDR=A0;
int PIR_VALUE=0;
int LDR_VALUE=0;
int threshold_value=300;
int BULB=8;
int BULB_VALUE=0;
long previous=0;
int threshold=400;
int day_threshold=300;

void setup()
{
```

```
Serial.begin(9600);
pinMode(PIR,INPUT);
pinMode(LDR,INPUT);
pinMode(BULB,OUTPUT);
pinMode(ipt,INPUT);
pinMode(opt,OUTPUT);
pinMode(kk,INPUT);
}

void loop()
{
if(millis()-(previous*1000)>3600000)
{
previous = millis()/1000;
digitalWrite(BULB,HIGH);
delay(10000);
LDR_VALUE=analogRead(LDR);
String sr;
if(LDR_VALUE > threshold)
{
sr="0";
}
else
{
sr="1";
}
String msgg="*"+sr;
Serial.println(msgg);
}
else
{
LDR_VALUE=analogRead(LDR);
PIR_VALUE=digitalRead(PIR);
if(PIR_VALUE==HIGH && LDR_VALUE > day_threshold)
{
BULB_VALUE= map(LDR_VALUE,0,1023,0,255);
analogWrite(BULB,BULB_VALUE);
}
else {
digitalWrite(BULB,LOW);
}
String sss=(BULB_VALUE==0)?"0":"1";
String msg="LDR_READING?"+String(LDR_VALUE)+"?PIR_
READING?"+String(PIR_VALUE)+"?BULB_READING?
"+String(BULB_VALUE)+"?BULB_STATE?"+sss+"?";
Serial.println(msg);
}
delay(5000);
}
```

VI. User Manaul

A. Components& their Uses

The different components used in this project is:

1) Passive Infrared (PIR) Sensor: It is used for sensing the motion of vehicles/persons around the streetlight. Working Input Voltage Range- 4.5V to 20V
2) Light Dependent Resistor(LDR) Sensor: It is used for detecting the intensity of Natural light available to decide if the bulbs need to be switched.
3) 1 channel 5V Realay: It is used to control the streetlight with respect to the data in the output pins. Working Input Voltage Range- 5V.
4) Arduino Board: The microcontroller used for reading the inputs from the sensor and transferring the data to the server.
5) NodeMCU ESP8266: The wifi enabled microcontroller used as the server. It can be used either as a WIFI STATION or as a ACCESS POINT or BOTH.

B. Setting up of Project

1) Assemble the circuit as per the configuration Diagram. Be sure to supply sufficient input voltage(Vcc) to the sensors and relay else they may behave strangely.
2) Upload the Server.ino code in NodeMCU, Client1.ino in 2nd NodeMCU and Client2.ino in Arduino UNO. Don't forget to make the necessary changes in WIFI.begin(SSID,PASSWORD) and sendNumber column.
3) Download the Android application from here.
4) Do some motion around any of the clients and you can observe the readings on the Serial Monitor and the Application.
5) You will receive the status of the streetlight on your phone number during periodic checks.

VII. Conclusion

The Street Light Controlling system works as desired. In the normal daylight when the intensity of light is enough and the streetlight does not turn on. During nights when the natural light intensity is not enough the streetlight becomes on when there is a motion in its neighbourhood. The same data can be easily seen in the Android application. There are periodic checks at the interval of 1 hour and we receive a text message on our phone. These are the main advantages of the Proposed System:

1) Automatic Switching of Street lights.
2) Maintenance Cost Reduction.
3) Reduction in CO2 emission.
4) Reduction of light pollution.
5) Wireless Communication.
6) Energy Saving.
7) Reduction of manpower.

Thus, with this proposed model we can leverage the IoT technology in true sense.

VIII. Video on Demonstration

Link to the video on demonstration: Demo Video