INTRODUCTION

As rapid change in technology always aims to serve the mankind, the expectation for living a simple yet advance life keeps on increasing [1].

Internet has become an important part of human's social life and educational life without which they are just helpless. The Internet of things (IoT) devices not only controls but also monitors the electronic, electrical and various mechanical systems which are used in various types of infrastructures. These devices which are connected to the cloud server are controlled by a single user (also known as admin) which are again transmitted or notified to the entire authorized user connected to that network [2-5].

Various electronics and electrical devices are connected and controlled remotely through different network infrastructures. Web browser present in laptop or smart phone or any other smart technique through which we can operate switches, simply removes the hassle of manually operating a switch. Now a day's although smart switches are available they proves to be very costly, also for their working we required additional devices such as hub or switch [3, 6].

As there is rapid change in wireless technology several connectivity devices are available in the market which solves the purpose of communicating medium with the device and the micro-controller. Starting from Bluetooth to Wi-Fi, from ZigBee to Z-wave and NFC all solve the purpose of communicating medium. RF and ZigBee are used to use in most wireless networks [4, 7].

In this project we have taken ESP8266-01 Wi-Fi module which is programmed through Arduino UNO to control various devices.

LITERATURE SURVEY

Home automation is a challenging one not only to the developer but also to the consumer. Developer has to choose the component as per the customer requirement. Due to all the customer demands are not equal hence they have to compromise with the existing products.

1. Through detailed study of “Home Automation Using Internet of Thing” proposed by Shopan Dey, Ayon Roy and Sandip Das, it is found that they have used Raspberry pi module to connect ESP8266-01 module to the internet.
2. Through this module they are controlling various devices through web page and also through android application [2]. K. Venkatesan and Dr. U. Ramachandraiah in their paper
3. have implemented Zigbee module in Arduino mega through which they are controlling devices. They have used various sensors for various purpose. Also they have provided real time notification, feedback on web-server in which customers can see what is
4. happening in their home [1]. With the help of logic gates, a Raspberry pi, 555 timer and flip-flop also the devices are controlled from web app. Paper proposed by Shashank Shiva Kumar Jha, Vishwateja Mudiam Reddy, Tapan Pokharna, Naresh Vinay shows how this is operated and controlled [3].
5. “Programmable Infrared Accessory Light Switch” by Warsuzarina Mat Jubadi and Normaziah Zulkifli shows how TV remote is used to control room light and other appliances. Here IR remote and one IR receiver is used and programmed in such a way that it stores the frequency of the existing remote and use them directly to control appliances [4].
6. So, here we introduce Arduino Uno with ESP8266-01 module. This is not only cost-effective but also proves to be the easiest one when it comes in term of programming and also implementation.

REQUIREMENT ANALYSIS

Proposed system specifications include hardware requirements and software requirements.

Hardware Requirement:

1. Temperature Sensor
2. 4 Channel Relay
3. Arduino Uno Board
4. ESP 8266 WIFI board
5. Desktop Machine as a Central Server
6. Android Smartphone as a Mobile client

Software Requirement:

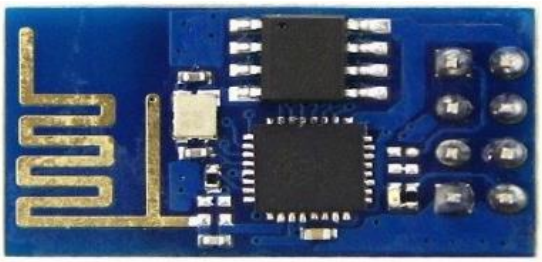
1. Arduino Software(IDE)
2. PostGreSql Database
3. RabbitMq Messaging System
4. Android Software(IDE)

Description:

Arduino Uno Board:

Arduino UNO is a microcontroller board based on ATmega328. It has 14 digital input/output pins of which 6 can be used as PWM output, 6 analog inputs. Arduino Uno can be programmed with Arduino software Arduino IDE (integrated development environment). The Atmel 8-bit AVRRISC-based microcontroller combines 32 kB ISP flash memory with read-while-write capabilities, 1 kB EEPROM, 2 kB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART. Serial communication interface is a feature in this board, including USB which will be used to load the programs from computer.

ESP2866:



ESP8266 Wi-Fi module is generally used to establish the wireless communication between the devices. But this module is not capable of 5-3v logic shifting and will require an external logic converter. Esp8266 having network connectivity is good for any computing system. And add to a system utility we can fetch any data from www. We can push data to cloud for storage, computation or monitoring. We need an external hardware that convert Wi-Fi data into data format that understood by common microcontroller like UAT, SPI, and I2C

Breadboard:

A Breadboard is used in learning of electronic used in connecting the components, testing it and pieces of bread well as the point holes in the lines to be used

Temperature Sensor:

It can measure temperature as well as humidity present in a room. Its range is less than 20 meters. It has a negative temperature coefficient (NTC) element and a humidity-sensitive element which is used to measure temperature between 0-50 degree Celsius.

4-Channel Relay:

4-Channel relay is connected to the Arduino Uno and its output is connected to the home appliances in a sequence as

(i) fan

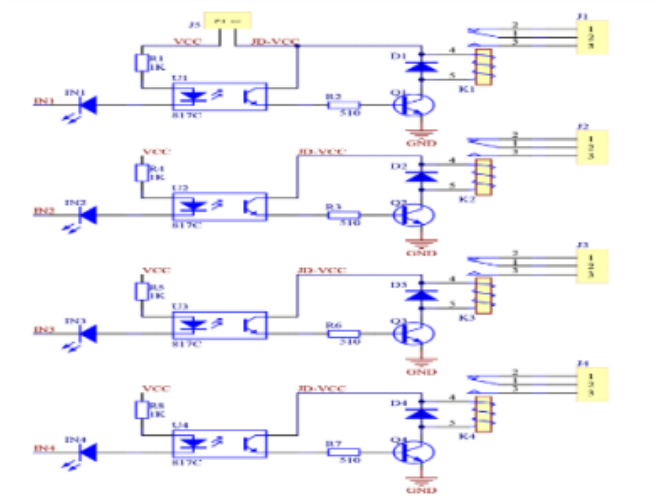
(ii) light

(iii) room-heater and

(iv) TV set.

Relay takes low current and voltage and triggers the switch which is connected to a high voltage.

4 input pins of relay are connected to Arduino which takes5V supply from it and can trigger upto 10A, 250V supply (Figure)



Arduino Studio:

The microcontrollers are typically programmed using a features from the programming languages C and C++. To install the Arduino software on windows following steps are useful [5].

Step 1- Download the Arduino software from Google.

Step 2- Install the software. Plug in your board and wait for Windows to begin its driver installation process.

Step 3- Open the control panel and open the system device manager.

Step 4- Connect Arduino Uno board to system through USB cable. After connecting select board and COM port in Arduino IDE.

Step 5- Develop an Arduino Code for sensors to cloud system in Arduino IDE, compile and upload the code in Arduino Uno board.

METHODOLOGY USED IN THE PROJECT

Software Design

We have used three different software for programming and controlling. IDE is open-source software which is not only used for writing programs but also for uploading code to Arduino. Android application for ESP8266-01 is available in the play store (Android smart phone) provides a platform to control different loads. This will only work if it is connected to the IP address and the port which is provided by the ESP8266-01 module as shown in below Fig. User can customize the application like load name, number of loads, its ON duration etc. For controlling ESP8266-01 through web browser or computer for real time notification EsPlorer is used

Figure. IP address and port for connection

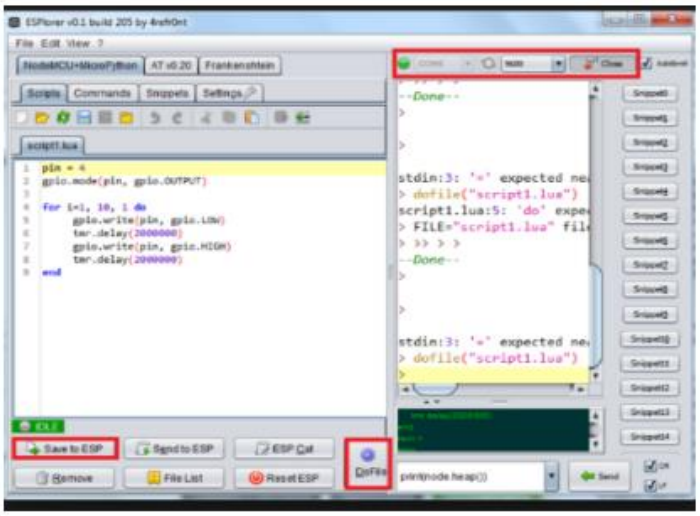


Figure ESPlorer in windows

METHODOLOGY USED IN THE PROJECT

Above Flow Chart figure is providing idea of overall operation of the system. Initially through Arduino programming, the system checks the modules as well as their connections. If any kind of error is detected by the system then it will indicate the ERROR status. If no error is found then the system will indicate the status OK and proceed for establishing the connection with the local Wi-Fi. Here the system will again check whether the ESP8266- 01 module is connected to the internet. If there is no connection then the system will indicate the ERROR status or else the display will show status SYSTEM ONLINE and show the IP address. The system will wait for the signal and switch the load accordingly after receiving the command and update the display.

To implements our home automation system, our system will be design as shown in below setup where we shall use Arduino Uno as a main controlling unit. And a four channel relay board to control electrical home appliance. And we will include a Wi-Fi module in our system to connect android and local Wi-Fi present in the home of user. We will test the experimental setup on various loads.

SYSTEM FLOW DIAGRAM

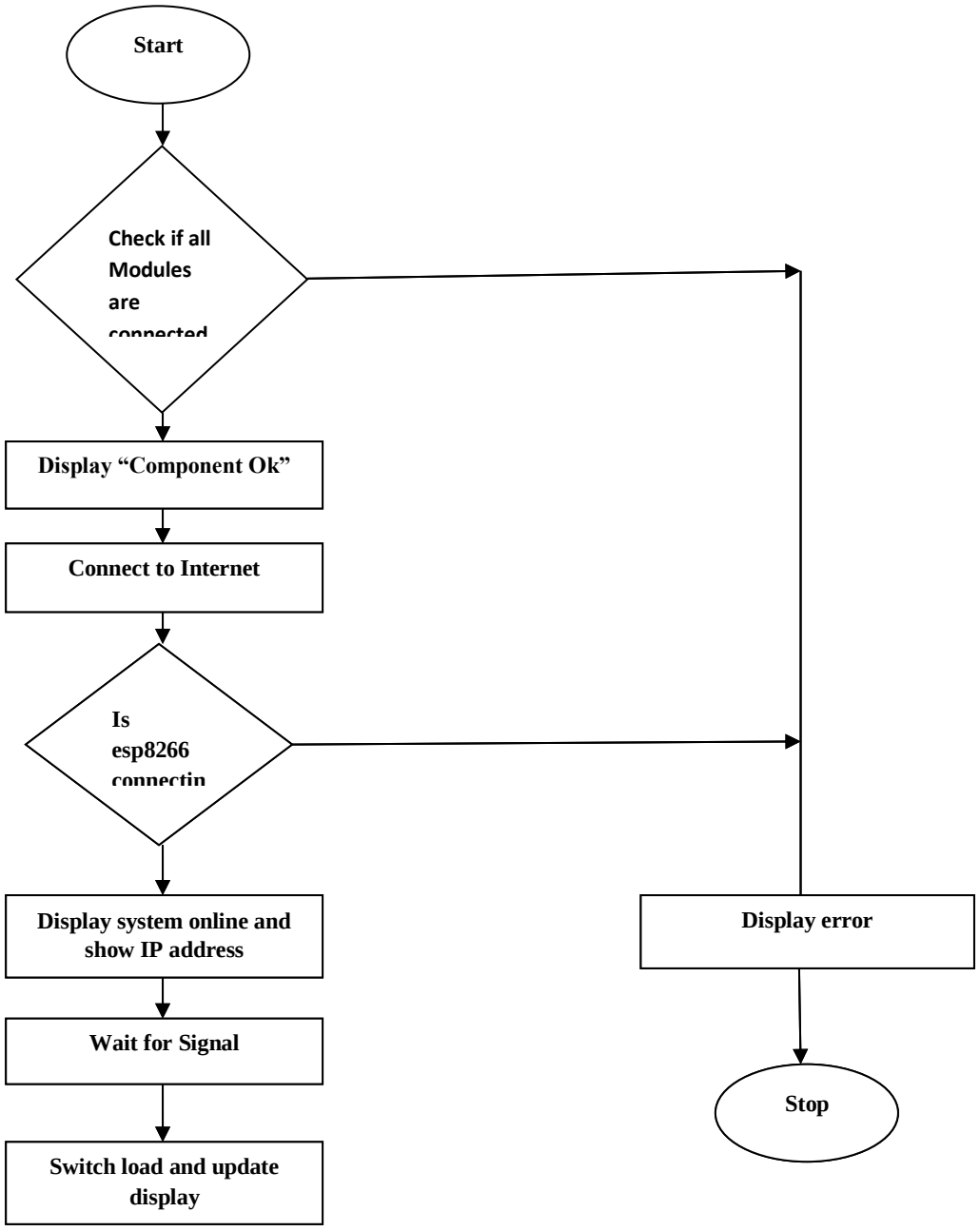


Fig. Flow of Program

IMPLEMENTATION AND CODING

ARDUINO UNO CODE

```
#include <SoftwareSerial.h>
int sensorPin = A0;   // select the input pin for ldr
int sensorValue = 0;  // variable to store the value coming from the sensor
SoftwareSerial nodeMCU(2,  3); // Rx,Tx .. Connection to NODEMCU
int transferStatus = 0;
void setup() {
  pinMode(9, OUTPUT); //pin connected to the relay
  Serial.begin(115200); //sets serial port for communication
  nodeMCU.begin(9600);
}

void loop() {

 if (nodeMCU.available())
  {
   char c = nodeMCU.read();
   Serial.println(c);

   switch (c)
    {
    case 'r':      // If received 'r'
```

```
   case 'R':      // or 'R'
     transferStatus = 1;
     break;
   case 's':      // If received 's'
   case 'S':      // or 'S'
     transferStatus = 0;
     break;
   }

  }

  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  Serial.println(sensorValue); //prints the values coming from the sensor on the screen
  nodeMCU.println(sensorValue); //

  if(sensorValue > 850) //setting a threshold value
  digitalWrite(9,HIGH); //turn relay ON
  else digitalWrite(9,LOW); //turn relay OFF

  if(transferStatus == 1)
  digitalWrite(9,HIGH); //turn relay ON
  else digitalWrite(9,LOW); //turn relay OFF

  delay(1000);
}
```

NODE MCU CODE

```
/*************************************************
  Adafruit MQTT Library ESP8266 Example
```

```
  Must use ESP8266 Arduino from:
    https://github.com/esp8266/Arduino

  Works great with Adafruit's Huzzah ESP board & Feather
  ----> https://www.adafruit.com/product/2471
  ----> https://www.adafruit.com/products/2821

  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Tony DiCola for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 ****************************************************/
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
#include <SoftwareSerial.h>

/************************* WiFi Access Point
*********************************/

#define WLAN_SSID       "ABCC"
#define WLAN_PASS       "ABC@123"

/************************* Adafruit.io Setup
*********************************/

#define AIO_SERVER      "192.168.1.6"
#define AIO_SERVERPORT  1883                   // use 8883 for SSL
#define AIO_USERNAME    ""
#define AIO_KEY         ""
```

```
/************ Global State (you don't need to change this!) ******************/

SoftwareSerial espserial(D6,D5); //RX,TX
// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;
// or... use WiFiFlientSecure for SSL
//WiFiClientSecure client;

// Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

/****************************** Feeds ***************************************/

// Setup a feed called 'photocell' for publishing.
// Notice MQTT paths for AIO follow the form: <username>/feeds/<feedname>
Adafruit_MQTT_Publish publishiotkitdata = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "homeautopub");

// Setup a feed called 'onoff' for subscribing to changes.
Adafruit_MQTT_Subscribe commandtoiotkit = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "homeautosub");

/*************************** Sketch Code ************************************/

// Bug workaround for Arduino 1.6.6, it seems to need a function declaration
// for some reason (only affects ESP8266, likely an arduino-builder bug).
void MQTT_connect();
```

```
void setup() {
 espserial.begin(9600);
  Serial.begin(115200);
  delay(100);

  Serial.println(F("Adafruit MQTT demo"));

  // Connect to WiFi access point.
  Serial.println(); Serial.println();
  Serial.print("Connecting to ");
  Serial.println(WLAN_SSID);

  WiFi.begin(WLAN_SSID, WLAN_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();

  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  // Setup MQTT subscription for onoff feed.
  mqtt.subscribe(&commandtoiotkit);
}

//uint32_t x=0;

void loop() {
  // Ensure the connection to the MQTT server is alive (this will make the first
```

```
  // connection and automatically reconnect when disconnected).  See the MQTT_connect
  // function definition further below.

char* dataRcv="";
  //int dataSend;
  MQTT_connect();

  // this is our 'wait for incoming subscription packets' busy subloop
  // try to spend your time here

  Adafruit_MQTT_Subscribe *subscription;
  while ((subscription = mqtt.readSubscription(1000))) {
    if (subscription == &commandtoiotkit) {
      //Serial.print(F("Got: "));
      //Serial.println((char *)commandtoiotkit.lastread);
      //espserial.write(F("Got: "));
      //espserial.write((char *)commandtoiotkit.lastread);

      dataRcv= (char *)commandtoiotkit.lastread;
      espserial.println(dataRcv);
        delay(100); //added on 05th March
      Serial.println(dataRcv);
      //espserial.write(dataRcv);

    }
  }

  // Now we can publish stuff!
  //Serial.print(F("\nSending photocell val "));
 //Serial.print(x);
 /// Serial.print("...");
  delay(100);
```

```
 if(espserial.available()>0){
   String data="";
       while(espserial.available()){
         char c=espserial.read();
         data=data+c;
         delay(5);
       }
       char* chr = strdup(data.c_str());
       //chr = strcat("{ \"Water Level\" : ", chr);
      // chr = strcat(chr," }");
       float no=data.toFloat();
       if(!publishiotkitdata.publish(chr)){
        Serial.println("sending failed");
       }else {
        //Serial.print("data send ok :");
       Serial.println(chr);
       }
       free(chr);
      // Serial.print("mqtt :");
      // Serial.println(data);
      // Serial.print("end :");
}

 // ping the server to keep the mqtt connection alive
 // NOT required if you are publishing once every KEEPALIVE seconds
 /*
 if(! mqtt.ping()) {
   mqtt.disconnect();
 }
 */
}
```

```
// Function to connect and reconnect as necessary to the MQTT server.
// Should be called in the loop function and it will take care if connecting.
void MQTT_connect() {
  int8_t ret;

  // Stop if already connected.
  if (mqtt.connected()) {
    return;
  }

  Serial.print("Connecting to MQTT... ");

  uint8_t retries = 5;
  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
       Serial.println(mqtt.connectErrorString(ret));
       Serial.println("Retrying MQTT connection in 5 seconds...");
       mqtt.disconnect();
       delay(5000);  // wait 5 seconds
       retries--;
       if (retries == 0) {
         // basically die and wait for WDT to reset me
         while (1);
       }
  }
  Serial.println("MQTT Connected!");
}
```

ANDRIOD STUDIO

MainActivity.java

```
package com.example.home;
```

```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.net.Uri;
import android.os.Handler;
import androidx.annotation.Nullable;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import org.eclipse.paho.android.service.MqttAndroidClient;
import org.eclipse.paho.client.mqttv3.IMqttActionListener;
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.IMqttToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.MqttPersistenceException;

import com.github.mikephil.charting.charts.LineChart;
```

```
import com.github.mikephil.charting.components.Legend;
import com.github.mikephil.charting.components.LimitLine;
import com.github.mikephil.charting.components.XAxis;
import com.github.mikephil.charting.components.YAxis;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;
import com.github.mikephil.charting.interfaces.datasets.ILineDataSet;
import com.github.mikephil.charting.utils.ColorTemplate;
import com.google.gson.Gson;
import com.pusher.client.Pusher;
import com.pusher.client.PusherOptions;
import com.pusher.client.channel.SubscriptionEventListener;

import java.util.Random;

public class MainActivity extends AppCompatActivity implements MqttCallback{

private LineChart mChart;

private Handler mHandler;

private static final float LIMIT_MAX_MEMORY = 12.0f;

public TextView intensity;
public ImageView img;
public static MqttAndroidClient client=null;

public static MqttConnectOptions options=null;

private Toast toast;
```

```
private Context context;
public  int flag=0,i=0;



@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

mHandler = new Handler();
mChart = (LineChart) findViewById(R.id.chart_view);
intensity = (TextView) findViewById(R.id.intensity);
img = (ImageView) findViewById(R.id.img);

        setupChart();
        setupAxes();
        setupData();
        setLegend();
//getData();
        //sensorData();
        //MqttConnection mc = new MqttConnection();
        //boolean ans = mc.Connection(this,"192.168.1.6","1883",)

}

public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.settings,menu);
return true;
    }

@Override
```

```
public boolean onOptionsItemSelected(MenuItem item) {
int id = item.getItemId();
//noinspection SimplifiableIfStatement
if (id == R.id.settings) {
            Intent intent = new Intent(this, Settings.class);
            startActivity(intent);
return true;
        }
return true;
    }



public void disconnect() {
        System.out.println( "disconnect...");
try {
client.unregisterResources();
//client.close(); // causes exceptions
client.disconnect();
client = null;
        } catch ( MqttException e ) {
            System.out.println("1disconnect exception: "+ e.toString() );
        } catch ( NullPointerException ne ) {
            System.out.println( "2disconnect exception:: "+ ne.toString() );
        }
    }

public void turnOn(View view){


        disconnect();
flag = 1;
img.setImageResource(R.drawable.lightbulb);
        sensorData();
```

```
    }

public void turnOff(View view){

        disconnect();
flag = 0;
img.setImageResource(R.drawable.no_light_bulb);
        sensorData();

    }




private void sensorData()
    {
options = new MqttConnectOptions();
options.setConnectionTimeout(0);
options.setMqttVersion(MqttConnectOptions.MQTT_VERSION_3_1);
        System.out.println("Server                                                    :
"+Preferences.getStringPreferences(getApplicationContext(),"server"));
if (Preferences.getStringPreferences(getApplicationContext(), "server") == null) {
            System.out.println("Server                                                :
"+Preferences.getStringPreferences(getApplicationContext(),"server"));
toast.makeText(getApplicationContext(),        "First        Configure        Server        ",
Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(MainActivity.this, Settings.class);
            startActivity(intent);

        }
```

```
else if (Preferences.getStringPreferences(getApplicationContext(),"server").equals("random"))
        {
            System.out.println("Null Server : "+Preferences.getStringPreferences(getApplicationContext(),"server"));
            getData();
        }
else
{
            System.out.println("Connecting to Server : "+Preferences.getStringPreferences(getApplicationContext(),"server"));
            System.out.println("Client : "+client);
try {
if (client != null) {
if (!client.isConnected()) {
                        System.out.println("Hey1");
                        Connection();
                    }
                } else {
                    Connection();
                }
            } catch (Exception e) {

                e.printStackTrace();
            }
//final String topic = Preferences.getStringPreferences(getApplicationContext(), "pubTopic");
            //System.out.println("pubTopic : "+topic);
            //pub(topic, "1");
}
```

```
    }
private boolean Connection() {
        String server = Preferences.getStringPreferences(getApplicationContext(), "server");
        String port = Preferences.getStringPreferences(getApplicationContext(), "port");
        String clientId = Preferences.getStringPreferences(getApplicationContext(),
"clientID");

//server="192.168.1.6";
        //port="1883";
        //clientId="ss";

System.out.println("Server : "+server);
        System.out.println("Port : "+port);
        System.out.println("Client ID : "+clientId);

try {
client = new MqttAndroidClient(this.getApplicationContext(), "tcp://"+ server + ":"+
port,
                clientId);

            System.out.println("Server URI : "+client.getServerURI());
            System.out.println("Client Id : "+client.getClientId());
            System.out.println("options : "+options);
            System.out.println("Hello Connection");
            System.out.println("Is Connected: "+client.isConnected());
            IMqttToken token = client.connect(options);
            token.setActionCallback(new IMqttActionListener() {
@Override
public void onSuccess(IMqttToken asyncActionToken) {
                System.out.println("Success : "+client.getClientId());
                Log.d("Success", "onSuccess");
```

```
//          Toast.makeText(DashBoard.this,          "Connection          successful",
Toast.LENGTH_SHORT).show();
                //subscribe();

if (flag == 1){

                String                                  topic                                  =
Preferences.getStringPreferences(getApplicationContext(),"pubTopic");
                System.out.println("Topic = "+topic);
boolean an = pub(topic,"r");

            }else{
                String                                  topic                                  =
Preferences.getStringPreferences(getApplicationContext(),"pubTopic");
boolean an = pub(topic,"s");

            }
            subscribe();

        }

@Override
public void onFailure(IMqttToken asyncActionToken, Throwable exception) {
            Log.d("Failure", "onFailure");
             Toast.makeText(MainActivity.this,          "Connection          failed",
Toast.LENGTH_SHORT).show();
        }
      });

    } catch (MqttException e) {
//e.printStackTrace();
System.out.println(e);
```

```
return false;
        } catch (Exception e) {
return false;
        }
return true;
    }
public static boolean pub(String topic, String payload) {
        System.out.println("Is Connected 1 : "+client.isConnected());
if (client != null) {
if (client.isConnected()) {
                System.out.println("Is Connected 2 : "+client.isConnected());
                MqttMessage message = new MqttMessage(payload.getBytes());
try {
client.publish(topic, message);
return true;
                } catch (MqttPersistenceException e) {
                    e.printStackTrace();
                } catch (MqttException e) {
                    e.printStackTrace();
                }
            }
        }
return false;
    }

void subscribe() {
        System.out.println("subscribed : "+client.getServerURI());
final String topic = Preferences.getStringPreferences(getApplicationContext(),
"subTopic");
int qos = 1;
try {
client.setCallback(MainActivity.this);
```

```
if (topic == null) {
            IMqttToken subToken = client.subscribe("SerialTopic", qos);
        }
        IMqttToken subToken = client.subscribe(topic, qos);

        subToken.setActionCallback(new IMqttActionListener() {
@Override
public void onSuccess(IMqttToken asyncActionToken) {
// successfully subscribed
                //  Toast.makeText(DashBoard.this, "Successfully subscribed to: " + topic,
Toast.LENGTH_SHORT).show();
}

@Override
public void onFailure(IMqttToken asyncActionToken,
                              Throwable exception) {
toast.makeText(getApplicationContext(), "Couldn't subscribe to: "+ topic,
Toast.LENGTH_SHORT).show();
            }
        });
      } catch (MqttException e) {
        e.printStackTrace();
      } catch (NullPointerException e) {
        e.printStackTrace();
      }
   }

@Override
public void messageArrived(String topic, MqttMessage message) throws Exception {

      String msg = new String(message.getPayload());
      System.out.println("message  : "+message);
```

```
//String a[] = msg.split("\n");
        //write the spliting logic as per data
i++;
if(i >1) {
int no = Integer.parseInt(msg);
            System.out.println("data = "+ msg);
            addEntry(no);
intensity.setText(msg);
        }

    }



@Override
public void connectionLost(Throwable cause) {

    }

@Override
public void deliveryComplete(IMqttDeliveryToken token) {

    }


private void setupChart() {
// disable description text
mChart.getDescription().setEnabled(false);
// enable touch gestures
mChart.setTouchEnabled(true);
// if disabled, scaling can be done on x- and y-axis separately
mChart.setPinchZoom(true);
```

```
// enable scaling
mChart.setScaleEnabled(true);
mChart.setDrawGridBackground(false);
// set an alternative background color
mChart.setBackgroundColor(Color.BLACK);
    }

private void setupAxes() {
        XAxis xl = mChart.getXAxis();
        xl.setTextColor(Color.WHITE);
        xl.setDrawGridLines(false);
        xl.setAvoidFirstLastClipping(true);
        xl.setEnabled(true);

        YAxis leftAxis = mChart.getAxisLeft();
        leftAxis.setTextColor(Color.WHITE);
        leftAxis.setAxisMaximum(1000f);
        leftAxis.setAxisMinimum(0f);
        leftAxis.setDrawGridLines(true);

        YAxis rightAxis = mChart.getAxisRight();
        rightAxis.setEnabled(true);

// Add a limit line
LimitLine ll = new LimitLine(LIMIT_MAX_MEMORY);
// ll.setLineWidth(2f);
        // ll.setLabelPosition(LimitLine.LimitLabelPosition.RIGHT_TOP);
        //ll.setTextSize(10f);
        //ll.setTextColor(Color.WHITE);
        // reset all limit lines to avoid overlapping lines
leftAxis.removeAllLimitLines();
        leftAxis.addLimitLine(ll);
```

```
// limit lines are drawn behind data (and not on top)
leftAxis.setDrawLimitLinesBehindData(true);
    }
private void setupData() {
        LineData data = new LineData();
        data.setValueTextColor(Color.WHITE);
```

TESTING

Test Case

Test Case ID	**005**
Test Case Summary	**Integration Testing.** Light Bulb to be made ON when ON Button Press by user on Mobile APP. Light Bulb to be made OFF when OFF Button press by user on Mobile App.
Prerequisites	1. LDR Sensor Connection established with Arduino UNO. 2. Power Supply of Arduino UNO and NODE MCU Unit to be made ON. 3. ESP8266 To be connected to WIFI Network. 4. Mobile Phone to be connected to WIFI Network
Test Procedure	1.Click on ON Button in Android App 2. Light Bulb should be ON 3. Click on OFF Button in Android App 4. Light Bulb should be OFF
Test Data	1. Data to be send by LDR sensor
Expected Result	Light Bulb to be made ON when ON Button Press by user on Mobile APP. Light Bulb to be made OFF when OFF Button press by user on Mobile App.
Actual Result	1.The system was working as expected
Status	Passed
Remarks	Sensor data correctly read by Arduino Uno and Published by ESP8266
Created By	Komal Garole
Date of Creation	20-Feb-20
Executed By	Arpit Umap
Date of Execution	5-Mar-20
Test Environment	Win-10; Arduino IDE

SCREEENSHOTS

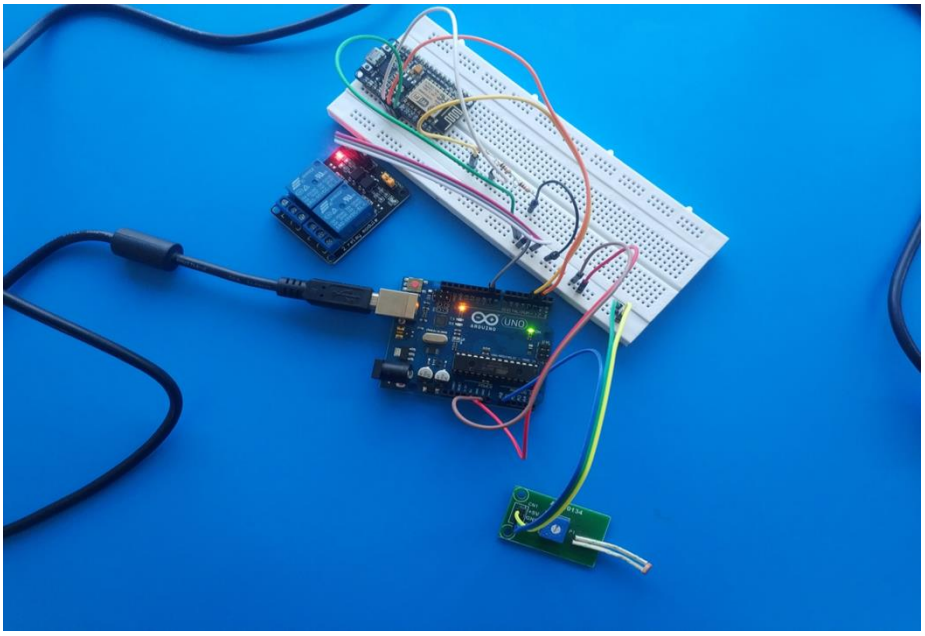


Fig. Module connections

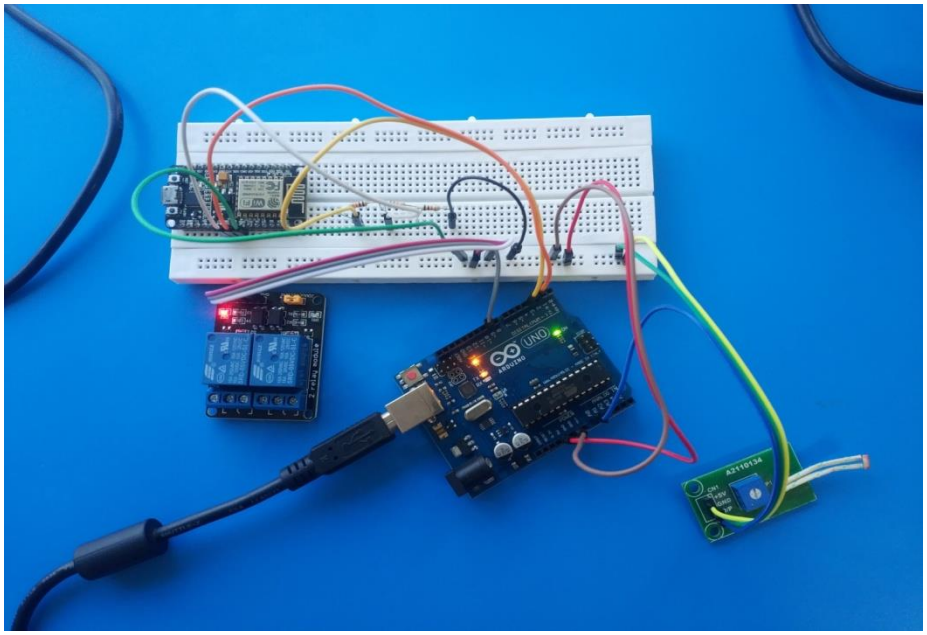


Fig. Module connections

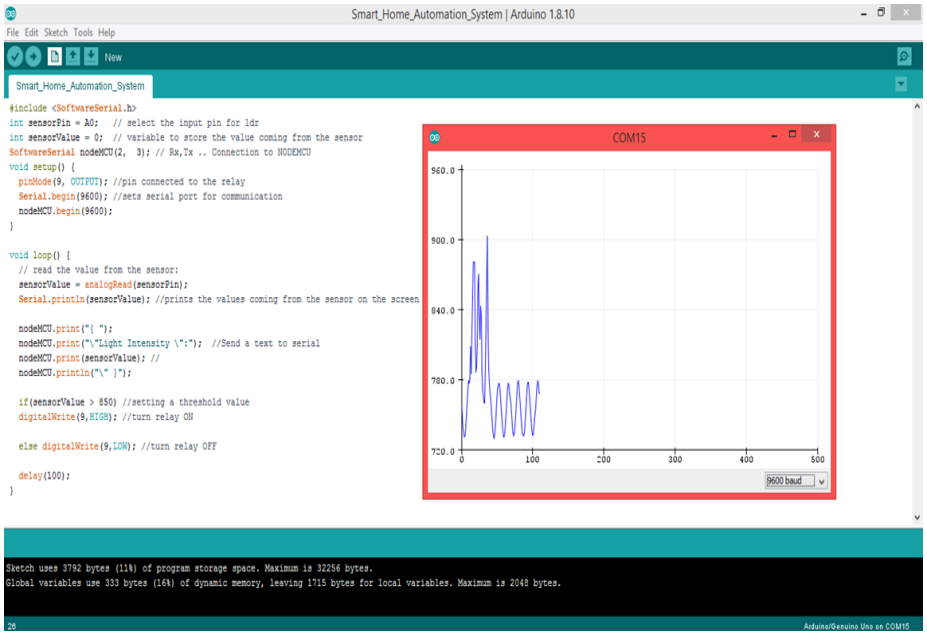


Fig. Screenshot of the Output screen

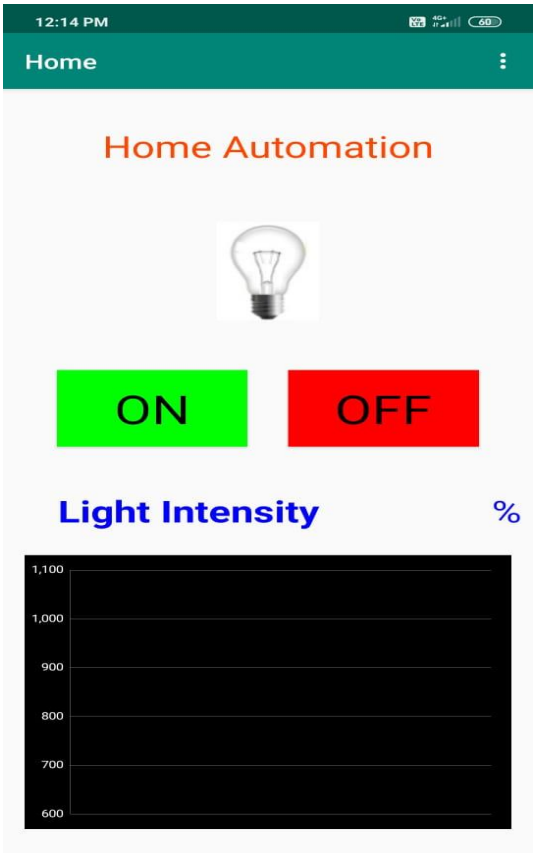


Fig. Screenshot of the App

REFERENCE

1. K. Venkatesan and Dr. U. Ramachandraiah, Networked Switching and Polymorphing Control of Electrical Loads with Web and Wireless Sensor Network, 2015 International Conference on Robotics, Automation, Control and Embedded Systems (RACE), Chennai, (2015), 1-9.

2. ShopanDey,Ayon Roy and SandipDas, Home Automation Using Internet of Thing , IRJET, 2(3) (2016),1965-1970.

3. VishwatejaMudiam Reddy, NareshVinay, TapanPokharna and Shashank Shiva Kumar Jha, Internet of Things Enabled Smart Switch, Thirteenth International Conference on Wireless and Optical Communications Networks (WOCN), Hyderabad, (2016),1-4

4. Warsuzarina Mat Jubadi and NormaziahZulkifli, Programmable Infrared Accessory Light Switch, International Conference on Intelligent and Advanced Systems, Kuala Lumpur,(2007), 1130-1134.

5. Shih-Pang Tseng, Bo-Rong Li, Jun-Long Pan, and Chia-Ju Lin, An Application of Internet of Things with Motion Sensing on Smart House, International Conference on Orange Technologies, Xian, (2014), 65-68.

6. Mandurano, Justin, and Nicholas Haber. House Away: A home management system, IEEE Long Island Systems, Applications and Technology Conference (LISAT), Farmingdale, NY, (2012), 1-4.

7. Zhen Bi, Smart home with ZigBee hardware simulation and performance evaluation, International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC), Shengyang, (2013), 2139-2142.

8. S. Karaca, A. Şişman and İ. Savruk, A low cost smart security and home automation system employing an embedded server and a wireless sensor network, International Conference on Consumer Electronics - Berlin (ICCE-Berlin), Berlin,(2016), 73-77.

9. T. Thaker, ESP8266 based implementation of wireless sensor network with Linux based web-server, Symposium on Colossal Data Analysis and Networking (CDAN), Indore, (2016), 1-5

10. Y. P. Zhang, T. Liu, Z. X. Yang, Y. Mou, Y. H. Wei and D. Chen, Design of remote control plug, 2015 IEEE International Conference on Applied Superconductivity and Electromagnetic Devices (ASEMD), Shanghai, (2015), 29-30.

11. Ss A. M. D. Celebre, A. Z. D. Dubouzet, I. B. A. Medina, A. N. M. Surposa and R. C. Gustilo, Home automation using raspberry Pi through Siri enabled mobile devices, International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM), Cebu City,(2015), 1-6.

12. R. Piyare, and S.R. Lee, Smart home-control and monitoring system using smart phone, The 1st International Conference on Convergence and its Application, 84, (2013) 83-86.

13. https://www.arduino.cc/en/main/arduinoBoardUno

14. https://www.northdoor.co.uk/iot-device-action

CONCLUSION

In this project our focused is on different process of operating or controlling electrical and electronic appliances remotely with the help of Arduino. This method of controlling such applications is referred to as automation. The experimental setup which we design will be having its focal point on controlling different home appliances providing 100% efficiency. Due to advancement in technology, Wi-Fi network is easily available in all places like home, Office Building and Industrial Building so proposed wireless network easily controlled using any Wi-Fi network. The wiring cost is reduced. Since less wiring is required for the switches. This also eliminates power consumption inside the building when the loads were in off conditions. This system is also platform independent allowing any web browser in any platform to connect ESP8266-01.The system is fully functional through android application known as “ESP8266 Wifi control”. Expecting delay to turn

ON is 3 sec and turn OFF is 2 sec for any load. For future use, the system can be further enhanced as: by adding speech recognition to the system