Minor project of IoT BASED SMART LED LAMP

Course Title: Skilling (WSN&IOT)

Course code:17EC3611

submitted by:

Sk. Mushtaq Hussain - 170040582

Submitted to:

Faculty Name: MR. N.L.S.P.SAIRAM

&

MR.S.VAMSEE KRISHNA

Department of ECE

(DEEMED TO BE UNIVERSITY)

KONERU LAKSHMAIAH EDUCATION FOUNDATION

(Deemed to be University, Estd. u/s. 3 of UGC Act 1956)

K L EDUCATIONAL FOUNDATION

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the project based report entitled “MINI WEATHER STATION” submitted by SK.MUSHTAQ HUSSAIN(170040582) to the Department of Electronic and Communication Engineering, Koneru Lakshmaiah Educational Foundation in partial fulfilment of the requirements for the completion of a project based in “WSN&IOT” in III B Tech I Semester, is a bonafide record of the work carried out by us during the academic year 2019 – 2020.

Project Supervisor	Head of the Department

ACKNOWLEDGMENTS

It is great pleasure for me to express my gratitude to our honorable President **Sri. Koneru Satya Narayana**, for giving the opportunity and platform with facilities in accomplishing the project based laboratory report.

I express the sincere gratitude to our principal **Dr.K.Subba Rao** his administration towards our academic growth.

I express sincere gratitude to our Coordinator and HOD-**Dr.Suman maloji** for her leadership and constant motivation provided in successful completion of our academic semester. I record it as my privilege to deeply thank for providing us the efficient faculty and facilities to make our ideas into reality.

I express my sincere thanks to our faculty in-charge and project supervisors **N.L.S.P.Sairam** sir and **S. Vamsee Krishna** sir his novel association of ideas, encouragement, appreciation and intellectual zeal which motivated us to venture this project successfully.

Finally, it is pleased to acknowledge the indebtedness to all Teaching and Non-teaching faculty those who devoted themselves directly or indirectly to make this project report success.

SUBMITTED BY :

170040582 - SK.MUSHTAQ HUSSAIN

TABLE OF CONTENTS

ABSTRACT

In this project, we handle the output pin of ESP32 for switching and dimming a DC LED lamp using a local Web server. For a local Web server, we do not need an Internet connection; we can handle everything over Wi-Fi. To make this work, we use additional circuitry, because LED being of high power cannot be directly controlled by ESP32**.**

In the circuit, we drive base voltage of BD139 using ESP32 pin 21. Base voltage controls the flow of potential between collector and emitter of BD139. This is an application of a transistor in amplifying region. Now, we can control voltage and, hence, brightness of an LED.

Power ESP32 using 5V supply (current not more than 500mA), or directly plug it in with the help of a USB or power bank. LED will be powered depending on its need—some are 12V, whereas others are 5V

INTRODUCTION:

ESP32 is a low power WiFi enabled microcontroller created and developed by Espressif Systems. The ESP32 is an advanced IoT microcontroller board possessing WiFi and Bluetooth Low Energy capabilities, as well as limited compatibility with the Arduino Core. This project demonstrating how to control light in our room remotely. So here we controlling led with WiFi via Blynk. Blynk is a Platform with iOS and Android apps to control Arduino, Raspberry Pi, and the likes over the Internet. It's a digital dashboard where you can build a graphic interface for your project by simply dragging and dropping widgets.

The 'Light controlling of ESP32 by WiFi ' project uses the ESP32 Development Board will be used to blink an LED at a specific timed interval, continuously. It is the required basic tutorial for any microcontroller board. And connect an LED to any of GPIO pin of ESP32

Smart lighting is broadly understood to cover the automation of lamp responses, such as dimming or on/off control to enhance user comfort and save energy.

Lighting has become smart and responsive, but when teamed with the emerging internet of things (IoT) it could support greater functionality if additional sensors were implemented.

The question as to how new generations of smarter lights can be introduced, cost-effectively, to existing networks needs to be addressed. The IoT is a fast-moving field, but owners fitting new LED lighting today are expecting to run lights for a long time before upgrading, to maximise their return on investment.

\This option uses the same app signature scheme as hardware secure boot, but unlike hardware secure boot it does not prevent the bootloader from being physically updated. This means that the device can be secured against remote network access, but not physical access. Compared to using hardware Secure Boot this option is much simpler to implement.

On first boot, the bootloader will generate a key which is not readable externally or by software. A digest is generated from the bootloader image itself. This digest will be verified on each subsequent boot.

Enabling this option means that the bootloader cannot be changed after the first time it is booted.

In this project, you're going to learn how to control the ESP32 As an example, we'll control two 12V lamps connected to a relay module. wall panel switches to physically control the lamps.

You’ll have a switch for each lamp. The switch changes the lamp’s state to the opposite of its current state. For example, if the lamp is off, press the wall switch to turn it on. To turn it off, you just need to press the switch again. Take a look at the figure below that illustrates how it works.

It can be easily attached to a wall with adhesive tap, without the need to make holes on the walls. Additionally, it is wireless, so you don’t need to worry about wiring and then hiding cables.

In order to operate this transistor in forward biased state, we have to apply current at its base and this base current must be greater than 1/10th of its collector current. Moreover, make sure to apply 5V at its base-emitter pin.

Once it’s operating in forward biased state, we can draw a maximum of 1.5A current between its Collector & Emitter. If maximum current i.e. 1.5A is flowing through a transistor then we can say it’s in Saturation Region.

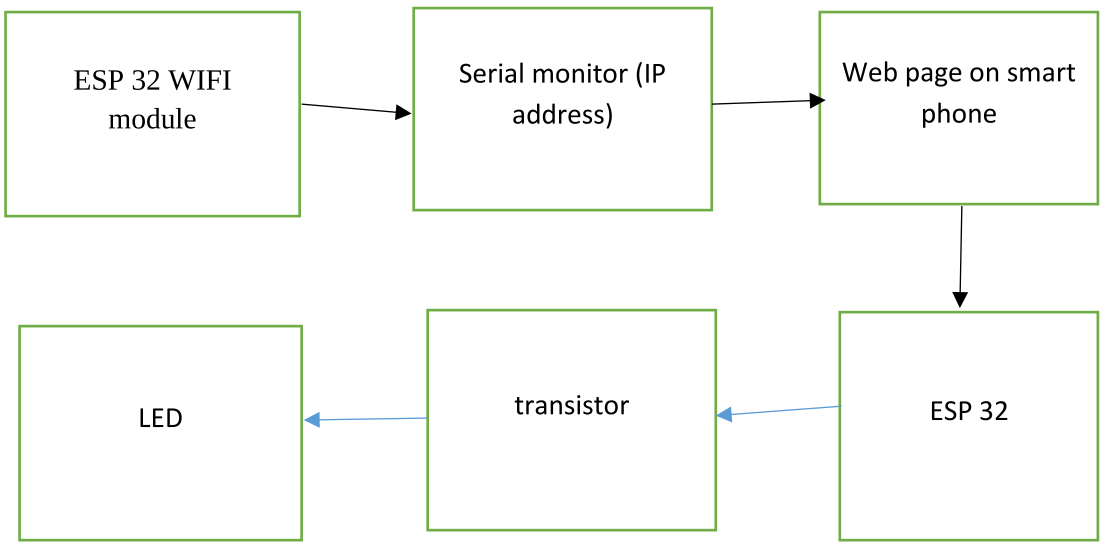
Normally, we can apply a maximum of 80V across Collector & Emitter.

When we remove base current transistor becomes fully off, this situation is called the cut-off region.

One best point about it is that it comes in a plastic package, which is that most medium power transistor available only in the metal package. This reduces its cost and since this package is not conductive it will not be affected by other circuits. Due to this feature, it is mostly used in amplifier applications.

So if you are searching for medium power NPN transistor in a plastic package than this will be the best choice for you.

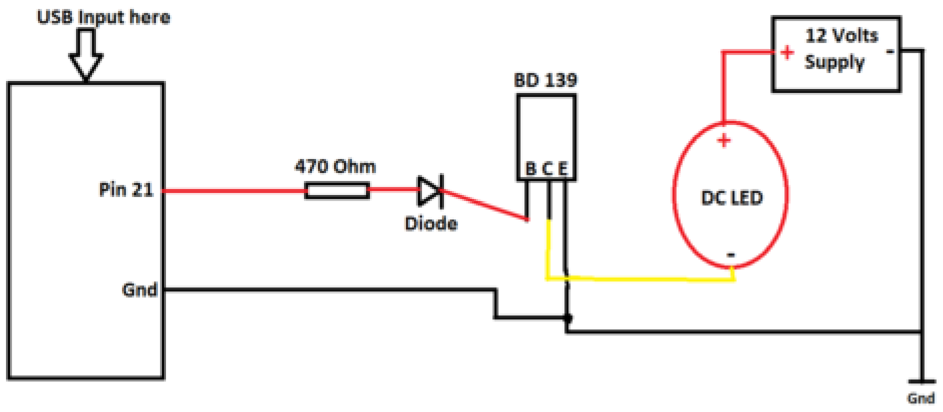
BLOCK DIAGRAM:



BLOCK DIAGRAM DESCRIPTION:

- The block diagram consits of Arduino Uno board,DHT 22 sensor,pressure sensor,LCD.
- Lcd is output device.Lcd display the temperature,humidity,light values.
- The **light sensor** is a passive devices that convert this "**light** energy" whether visible or in the infra-red parts of the spectrum into an electrical signal output.
- **Barometer** sensor can be used to measure temperature and atmospheric pressure accurately.
- The **Temperature(DHT22)** sensor is a basic, low-cost digital temperature and humidity sensor.

CIRCUIT DIAGRAM:



COMPONENTS REQUIRED:

S.NO	Components Name	Quantity
1	ESP32 Node MCU	1
2	USB Type C cable	1
3	20W 12V DC LED with heat-sink	1
4	NPN BD-139 transistor	1
5	470-ohm resistor	1
6	Regulated 12V supply from a battery eliminator or a charger with 12V 1A rating	1
7	Some wires and a breadboard	Required

COMPONENTS EXPLANATION

ESP32 Node MCU

ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series employs a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations and includes in-built antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules. ESP32 is created and developed by Espressif Systems, a Shanghai-based Chinese company, and is manufactured by TSMC using their 40 nm process. It is a successor to the ESP8266 microcontroller.

USB-C cables

It can carry significantly more power, so they can be used to charge larger devices like laptops. They also offer up to double the transfer speed of **USB** 3 at 10 Gbps. While connectors are not backwards compatible, the standards are, so adapters can be used with older devices.

20W 12V DC LED with heat-sink

20W White LED with Heatsink Cooler Fan Unit,Ready to Use Setup for Home Lighting Free Shipping,Prefabricated Unit,Heavy Duty,as per Specifications Nots :USE 12V 2 AMP POWER SUPPLY /SMPS

BD139

It is a Bipolar NPN transistor, it is mounted in the SOT-32 plastic package.

It is used as RF Amplifiers.
It is used in switching circuits.
It is used in amplification circuits.
It is used in audio amplifiers.
It is also used in Load driver circuits**.**

470 Ohm Through Hole Resistor

A passive device that resists the flow of electricity. This resistor will provide 470 Ohms of resistance wherever it is paced and will handle 1/4 watts. Use these low value resistors for voltage dividers and where you need to keep the current flow as high as possible

Regulated 12V supply

Lithium-ion charges similarly to lead acid and you can also use the power supply but exercise extra caution. Check the full charge voltage, which is commonly 4.20V/cell, and set the threshold accordingly. Make certain that none of the cells connected in series exceeds this voltage. (The protection circuit in a commercial pack does this.) Full charge is reached when the cell(s) reach 4.20V/cell voltage and the current drops to 3 percent of the rated current, or has bottomed out and cannot go down further. Once fully charged, disconnect the battery. Never allow a cell to dwell at 4.20V for more than a few hours.

Writing Sketches:

Programs written using ARDUINO Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the ARDUINO Software (IDE), including complete error messages and other information. The bottom right hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

✓ Verify

Checks your code for errors compiling it.

➜ Upload

Compiles your code and uploads it to the configured board.

See uploading below for details.

Note: If you are using an external programmer with your board, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"

Additional commands are found within the five menus: File, Edit, Sketch, Tools, and Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

New
Creates a new sketch.

Open
Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the File | Sketchbook menu instead.

Save
Saves your sketch.

Serial Monitor
Opens the serial monitor.

UPLOADING:

Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus. The boards are described below. On the Mac, the serial port is probably something like /dev/tty.usbmodem241 (for a Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board), or /dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be

/dev/ttyACMx, /dev/ttyUSBx or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the File menu. Current ARDUINO boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The ARDUINO Software (IDE) will display a message when the upload is complete, or show an error.

When you upload a sketch, you're using the ARDUINO boot-loader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The boot-loader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The boot-loader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

Libraries:

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compiles the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its #include statements from the top of your code.

There is a list of libraries in the reference. Some libraries are included with the ARDUINO software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these instructions for installing a third-party library.

Third-Party Hardware:

Support for third-party hardware can be added to the hardware directory of your sketchbook directory. Platforms installed there may include board definitions (which 2appear in the board menu), core libraries, boot-loaders, and programmer definitions. To install, create the hardware directory, then unzip the third-party platform into its own sub-directory. (Don't use "ARDUINO" as the sub-directory name or you'll override the built-in ARDUINO platform.) To uninstall, simply delete its directory.

For details on creating packages for third-party hardware, see the Arduino IDE 1.5 3rd party Hardware specification.

Serial Monitor:

Displays serial data being sent from the Arduino or Genuino board (USB or serial board). To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down that matches the rate passed to Serial. begin in your sketch. Note that on Windows, Mac or Linux, the Arduino or Genuino board will reset (rerun your sketch execution to the beginning) when you connect with the serial monitor.

You can also talk to the board from Processing, Flash, MaxMSP, etc. (see the interfacing page for details).

Preferences:

Some preferences can be set in the preferences dialog (found under the Arduino menu on the Mac, or File on Windows and Linux). The rest can be found in the preferences file, whose location is shown in the preference dialog.

Language Support:

Since version 1.0.1, the Arduino Software (IDE) has been translated into 30+ different languages. By default, the IDE loads in the language selected by your operating system. (Note: on Windows and possibly Linux, this is determined by the locale setting which controls currency and date formats, not by the language the operating system is displayed in.)

If you would like to change the language manually, start the Arduino Software (IDE) and open the Preferences window. Next to the Editor Language there is a dropdown menu of currently supported languages. Select your preferred language from the menu, and restart the software to use the selected language. If your operating system language is not supported, the Arduino Software (IDE) will default to English.

You can return the software to its default setting of selecting its language based on your operating system by selecting System Default from the Editor Language drop-down. This setting will take effect when you restart the Arduino Software (IDE). Similarly, after changing your operating system's settings, you must restart the Arduino Software (IDE) to update it to the new default language.

Boards:

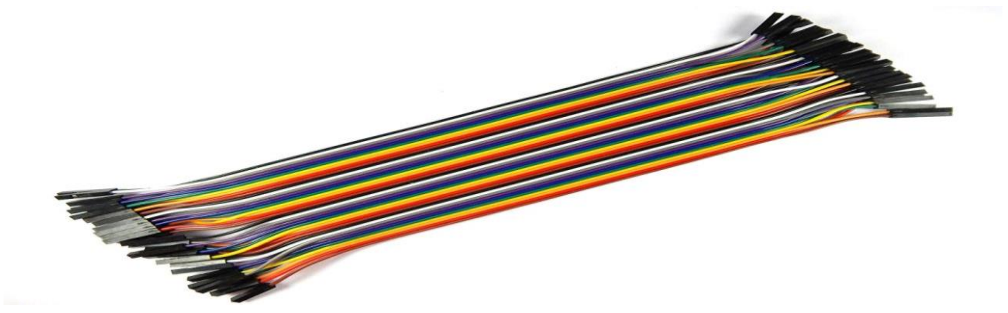
The board selection has two effects: it sets the parameters (e.g. CPU speed and baud rate) used when compiling and uploading sketches; and sets and the file and fuse settings used by the burn bootloader command. Some of the board definitions differ only in the latter, so even if you've been uploading successfully with a particular selection you'll want to check it before burning the bootloader. You can find a comparison table between the various boards here.

Arduino Software (IDE) includes the built in support for the boards in the following list, all based on the AVR Core. The Boards Manager included in the standard installation allows to add support for the growing number of new boards based on different cores like Arduino Due, Arduino Zero, Edison, Galileo and so on.

JUMPERWIRES:

A jump wire (also known as jumper wire, or jumper) is an electrical wire, or group of them in a cable, with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

Individual jump wires are fitted by inserting their "end connectors" into the slots provided in a breadboard, the header connector of a circuit board, or a piece of test equipment.



WORKING PRINCIPLE:

- The brightness of led lamp is controlled using esp32 pin 21
- In the web page generated by Arduino, we have 6 buttons from “off” to level 5 ,by pressing these buttons we can control the brightness of the led
- The brightness of led is controlled by bd 139 transistor
- Everything is controlled over Wifi, when we change the inputs the voltage given to the led is changed hence brightness is changed
- USB Type C cable to program ESP32 from a laptop or PC—most Android phones use this type of cable.
- 20W 12V DC LED with heat-sink, which will act as the lamp to control over Wi-Fi.
- NPN BD-139 transistor to control brightness or, technically, voltage supplied to LED. We will drive its base voltage through ESP32.
- 470-ohm resistor for BD-139 base and IN4007 diode to protect the circuit from any reverse voltages from 12V supply.
- Regulated 12V supply from a battery eliminator or a charger with 12V 1A rating

CODE:

```
#include <WiFi.h>                                               // This library contains the
Wifi function to run on ESP-32

#define LEDC_CHANNEL_0     0                                         // In these 4 lines
we have defined the PWM details for LED
#define LEDC_TIMER_13_BIT  13
#define LEDC_BASE_FREQ     5000
#define LED_PIN            21

#define min(a,b) ((a)<(b)?(a):(b));                                  // Defining Min
Function

const char* ssid     = "nagagowtham";
const char* password = "8331001834";

WiFiServer server(80);



int brightness = 0;

void ledcAnalogWrite(uint8_t channel, uint32_t value, uint32_t valueMax = 255)        //
function for analogWrite in Arduino
{
  uint32_t duty = (8191 / valueMax) * min(value, valueMax);
  ledcWrite(channel, duty);
}
```

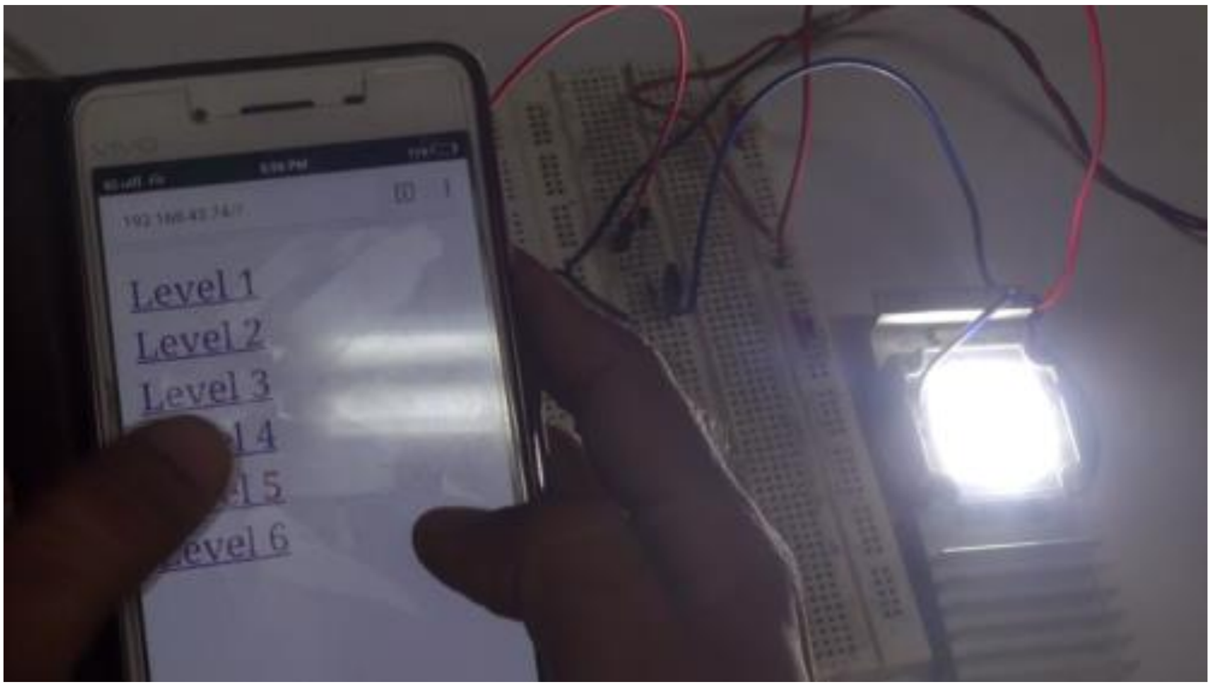
```
void setup()
{
   Serial.begin(115200);                                        //Baud Rate of ESP-32
   delay(10);

   Serial.println();
   Serial.print("Connecting to ");
   Serial.println(ssid);
   WiFi.begin(ssid, password);
   while (WiFi.status() != WL_CONNECTED)
   {
      delay(500);
      Serial.print(".");
   }
   Serial.println("");
   Serial.println("WiFi connected.");
   Serial.println("IP address: ");
   Serial.println(WiFi.localIP());
   server.begin();                                          // Start the server
   ledcSetup(LEDC_CHANNEL_0, LEDC_BASE_FREQ, LEDC_TIMER_13_BIT);
// PWM setup
   ledcAttachPin(LED_PIN, LEDC_CHANNEL_0);
}
int value = 0;
void loop(){
 WiFiClient client = server.available();                               // listen for any
incoming client

  if (client) {                                          // if you get any client,
   Serial.println("New Client.");                                     // print a message on
the serial monitor
   String currentLine = "";                                        // make a String to hold
incoming data from the client
   while (client.connected())                                      // loop while the client's
connected
   {
     if (client.available())                                    // if there's bytes to read from
the client
     {
      char c = client.read();                                     // read a byte, then
      Serial.write(c);                                        // print it out on the serial
monitor
      if (c == '\n')                                        // if the byte is a newline
character
      {
        if (currentLine.length() == 0)                               // if the current line is
blank, you got two newline characters in a row.
                                                         // that's the end of the client HTTP
request, so send a response:
         {
```

```
      client.println("HTTP/1.1 200 OK");                    // HTTP headers
always start with a response code (e.g. HTTP/1.1 200 OK)
      client.println("Content-type:text/html");             // and a content-
type so the client knows what's coming,
      client.println();                                 // then a blank line:
                                                    // the content of the HTTP response
follows the header:
      client.print("<a href=\"/A\">OFF</a><br>");
      client.print("<a href=\"/B\">Level 1</a><br>");
      client.print("<a href=\"/C\">Level 2</a><br>");
      client.print("<a href=\"/D\">Level 3</a><br>");
      client.print("<a href=\"/E\">Level 4</a><br>");
      client.print("<a href=\"/F\">Level 5 Full</a><br>");
      client.println();                                 // The HTTP response ends
with another blank line:
      break;                                          // break out of the while loop:
     }
     else
     {
      currentLine = "";                                   // if you got a newline, then
clear currentLine:
     }
    }
    else if (c != '\r')                                 // if you got anything else but a
carriage return character,
    {
     currentLine += c;                                    // add it to the end of the
currentLine
    }
    if (currentLine.endsWith("GET /A"))                           // Check to see if
the client request was "GET /A" or "GET /B" and so on...:
    {
     brightness = 0;
    }
    if (currentLine.endsWith("GET /B"))
    {
     brightness = 50;
    }
    if (currentLine.endsWith("GET /C"))
    {
     brightness = 100;
    }
    if (currentLine.endsWith("GET /D"))
    {
     brightness = 150;
    }
    if (currentLine.endsWith("GET /E"))
    {
     brightness = 200;
    }
```

```
        if (currentLine.endsWith("GET /F"))
        {
          brightness = 255;
        }
        ledcAnalogWrite(LEDC_CHANNEL_0, brightness);                                       // the
analog value of brightness is written on the pin
      }
    }
    client.stop();                                                   // close the connection:
    Serial.println("Client Disconnected.");
  }
}
```

OUTPUTS AND RESULTS:

Therefore by selecting a particular brightness on the web page we can change the brightness of the led as shown below:



ADVANTAGES

- This project is great for those who need DC lamps to study or to do any other activity.
- Longevity
- They light up faster
- Cost Effectiveness
- Durability
- Less Risk of Injury

DISADVANTAGES

- High initial cost
- Temperature Sensitive
- Different Colour Perception
- Voltage Sensitive
- Electric Polarity
- Eye Strain

CONCLUSION & FUTURE SCOPE:

We are concluding that this project helps us to know and control the LED using WiFi module and controlled using smartphone.

The components used in the project, like Arduino and sensors are slowly becoming an indispensable part of our daily routines. So, it is only fitting that we use them to improve efficiency in every walk of life.

Despite their high initial costs, they are a viable option as they drastically reduce the power consumption. They will aid in further saving of energy and reduction in operational costs.

REFERNCES:

1. https://create.arduino.cc/projecthub/igorF2/arduino-uno-mini-weather-station-31b555?ref=tag&ref_id=iot&offset=9
2. https://circuitdigest.com/microcontroller-projects/arduino-lcd-interfacing-tutorial