,,,

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
ENGENHARIA MECATRÔNICA
TÓPICOS ESPECIAIS EM SISTEMAS EMBARCADOS

RELATÓRIO DE IMPLEMENTAÇÃO - CONTROLADOR DE LUMINOSIDADE

ATYSON JAIME DE SOUSA MARTINS

Natal-RN
MARÇO/2021

ATYSON JAIME DE SOUSA MARTINS

RELATÓRIO DE IMPLEMENTAÇÃO - CONTROLADOR DE LUMINOSIDADE

Relatório apresentado à disciplina de tópicos especiais em sistemas embarcados, correspondente à avaliação da 1ª unidade do semestre 2020.2 do curso de Engenharia Mecatrônica da Universidade Federal do Rio Grande do Norte, sob orientação do **Prof. Dr. Victor Araújo Ferraz.**

Professor: Victor Araújo Ferraz.

Natal-RN
MARÇO/2021

Lista de Figuras

Sumário

1 INTRODUÇÃO

Controladores de luminosidade são comumente utilizados por grandes industrias alimentícias, como também, produtores de animais e hortifrúti que desejam ou precisam manter seu estabelecimentos a uma certa quantidade de calor e temperatura independe da hora do dia, como pode ser visualizada na figura 1.

O uso de dispositivos controláveis por software como microcontroladores, auxiliam no processo de implementação desses tipos de controladores, possibilitando a implementação de centrais a distancia, visualização em tempo real da porcentagem ou quantidade de iluminação na região, entre outras inúmeras possibilidades.

O presente trabalho tem por objetivo exibir a metodologia de projeto para desenvolvimento um controlador de luminosidade, capaz de ser controlado através de um servidor web criado pelo microcontrolador presente no Esp 32.

Ademais, em sua pagina web criada, sera possível visualizar a porcentagem de luminosidade, bem como, apresentará algumas funcionalidades possíveis.

Figura 1: Estufa

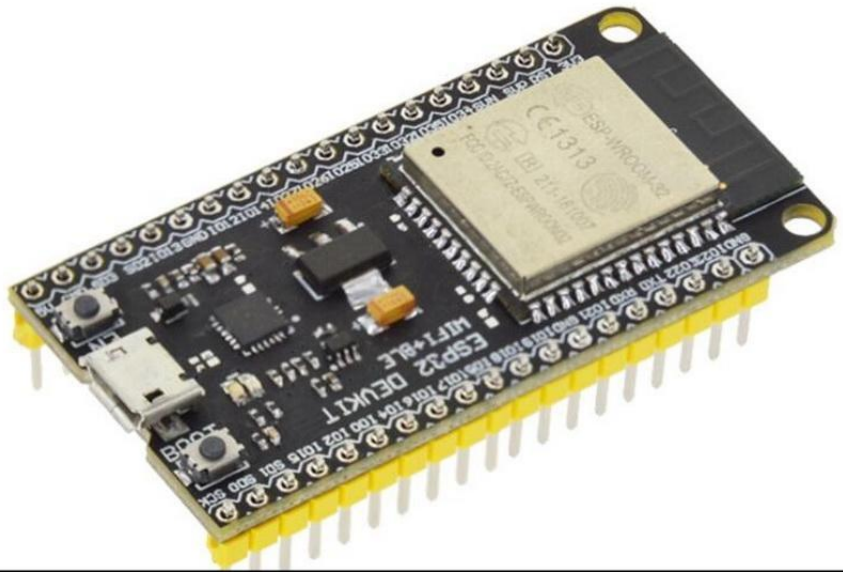
Fonte: Imagem disponibilizada no google.

2 DESENVOLVIMENTO

2.1 ESP32-wroom-32

ESP32 ou Esp-wroom-32 é uma placa de desenvolvimento criada pela empresa (*Espressif Systems*), possui como sua antecessora o ESP8266. Este pequeno componente é bastante difundido em aplicações de automações no geral, pois em sua placa além de apresentar módulo de comunicação Wi-Fi, apresenta um sistema com processador Dual Core, Bluetooth híbrido e múltiplos sensores embutidos, tornando a construção de sistemas mais fácil e simplificado. Para esse projeto, utilizamos o modulo de comunicação wi-fi, a posteriori será melhor explicado sobre esse modulo. Outrassim, para conseguir desenvolver e programar o esp-32 foi usada a IDE do Arduino.

Figura 2: Placa de Desenvolvimento Esp-32



Fonte: Imagem disponibilizada no google

2.1.1 Wi-Fi

Algumas característica desse modulo são as seguintes:

- IEEE 802.11 b/g/n (2.4 GHz até 150 Mbps;
- WPA/WPA2/WPA2-Enterprise e WPS;
- AMPDU, HT40 e QoS;
- Suporta vários modos (Infrastructure Station, SoftAP, Station+AP, Promíscuo (Sniffer));

Como a IDE utilizada foi a do Arduino, precisamos importar a biblioteca wifi como as funções que iremos utilizar. Desse modo, iremos configurar o código da seguinte maneira:

```
#include <WiFi.h>

// Wifi config
char ssid[] = "-----";   //  Nome da sua rede
char pass[] = "-----";   // Senha da sua rede
int keyIndex = 0;
int status = WL_IDLE_STATUS;
WiFiServer server(80);

void setup(){
  Serial.print("Tentando conexão a rede: ");
  Serial.println(ssid);
  while (status != WL_CONNECTED) {
    status = WiFi.begin(ssid, pass);
    Serial.print("Conectando...");
    delay(10000);
  }
  Serial.print("Conectado");
  server.begin();
  printWifiStatus();
}
```

A função *printWifiStatus()* tem como objetivo nos informar que a conexão foi bem sucessividade e mostrar qual é o ip que devemos colocar em nosso navegador para ter acesso a pagina web criada pelo esp 32.

```
void printWifiStatus() {
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
  // print where to go in a browser:
  Serial.print("Por favor, acesse a pagina através do endereço: http://");
  Serial.println(ip);
};
```

As funcionalidades geradas pelos esp eram controladas através de requisições GET feitas a partir do servidor Web criado. Sendo assim, quando o esp recebe essa requisição, o

transforma em alguma função interna dela. No caso desse projeto, poderia ocorrer quatro diferentes funções: Ligar LED, Desligar LED, Aumentar Luminosidade e Reduzir Luminosidade.

2.1.2 PWM

Como precisamos controlar a intensidade que a luz é emitida, utilizamos o modulo PWM do microcontrolador. O PWM funciona modulando o ciclo ativo (duty cicle) de uma onda quadrada. O conceito de funcionamento é simples. O controlador (fonte de tensão com PWM) entrega uma série de pulsos, gerados em intervalos de igual duração, que pode ser variada. Quanto mais largo o pulso, maior a quantidade de corrente fornecida à carga. Assim, precisamos primeiramente fazer algumas configurações:

```
// PWM LED
#define LED_PIN 14    // Definição da porta do LED
int freq = 5000;      // Frequencia utilizada para onda quadrada
int ledChannel = 0;   // Canal do PWM
int resolution = 8;   // Resolução do PWM
int dutyCycle = 0;    // Inicialização do dutyCycle

void setup() {
  // Conf PWM
  ledcSetup(ledChannel, freq, resolution);
  ledcAttachPin(LED_PIN, ledChannel);
  ledcWrite(ledChannel, dutyCycle);
}
```

2.1.3 Entradas Analógicas

Sabendo que os valores recebidos como dado do sensor não são digitais, ou seja, 0v e 5v. Precisamos utilizar as estradas analógicos do esp 32 para conseguir ter a precisão melhor. Para tal, precisamos fazer a seguinte configuração no código:

```
//Analog Input
#define ANALOG_PIN_0 32 // Define a entrada analógica
int analog_value = 0;  // Inicializa a entrada como zero
int LDR_value_porcentagem = 0;  // Transforma valor recebido em porcentagem

void loop(){
    analog_value = analogRead(ANALOG_PIN_0); // Função para receber o dado analógico;
    LDR_value_porcentagem = map(analog_value, 0, 2870, 0, 100); // Função para mapiar o valor
    //e conseguir descifrar a porcentagem de luminosidade;
}
```

2.2 Sensor

O sensor utilizado para o desenvolvimento desse respectivo trabalho é chamado de LDR, como pode ser visto na imagem 3, o Sensor de Luminosidade LDR (Light Dependent Resistor) é um componente cuja resistência varia de acordo com a intensidade da luz. Quanto mais luz incidir sobre o componente, menor a resistência. Assim, é a partir dele que conseguirmos saber qual a intensidade da luz (no nosso caso um led) sobre o ambiente e, desse modo, conseguir discernir entre ligar ou desligar a luz, como também, aumentar ou diminuir a intensidade da mesma.

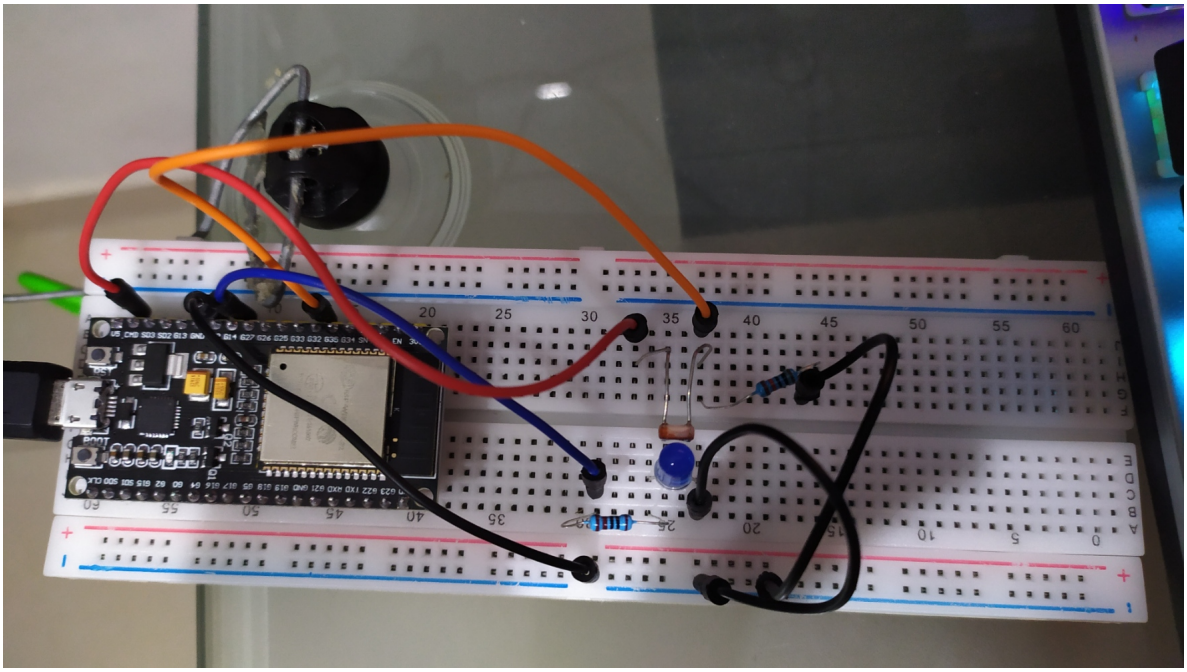
Figura 3: Sensor de Luminosidade LDR

Fonte: Imagem disponibilizada no google

3 RESULTADOS

Nessa secção apresentarei os resultado conseguidos com o controlador de luminosidade. A figura 4 mostra como ficou o resultado do projeto montado em uma protoboard. Na imagem podemos visualizar o esp-32, o sensor LDR e o led azul representando a luz a ser medida.

Figura 4: Modelo montado em protoboard



Fonte: Autoria própria

Em seguida, podemos ver o resultado da pagina html criada quando o usuário se conecta e acessa através do ip apresentado pelo esp-32. O código para construção da pagina é visível na estrutura mostrada na figura 5.

Figura 5: Codigo Html na IDE Arduino

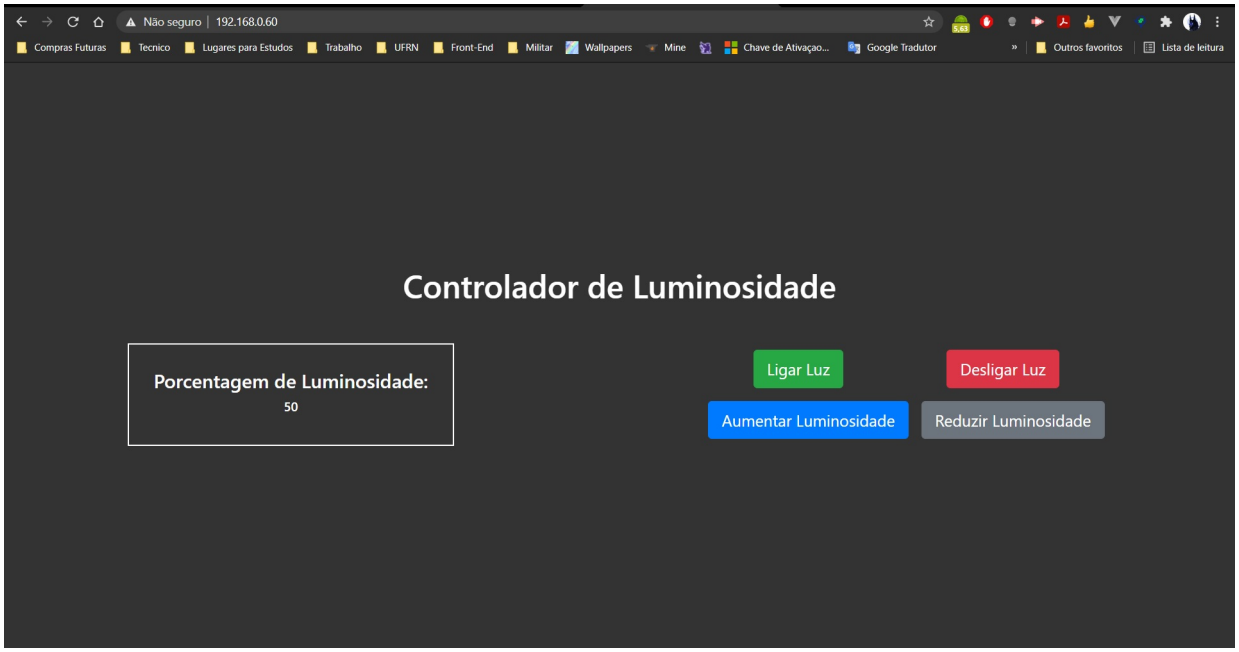
```
client.println("HTTP/1.1 200 OK");
client.println("Content-type:text/html");
client.println();

// Estrutura HTML
client.print("<html lang='pt-br'>");
client.print("<head><meta charset='UTF-8'><meta http-equiv='X-UA-Compatible' content='IE=edge'>");
client.print("<meta name='viewport' content='width=device-width, initial-scale=1.0'><title>Controlador de Luminosidade</title>");
client.print("<link rel='stylesheet' href='https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css' integrity='sha384-B0vP5xmATw1+K9KRQjQERJvTumQW0nPEzvF6L/Z6nronJ3oUOFUFpCjEUQouq2+l' crossorigin='anonymous'>");
client.print("<style>html,body {height: 100%;background-color: #333;}</style></head>");
client.print("<body><div class='w-100 h-100 d-flex flex-column justify-content-center align-items-center'>");
client.print("<h1 class='text-white text-center m-4'>Controlador de Luminosidade</h1><div class='d-flex w-100 justify-content-around flex-wrap'>");
client.print("<div class='text-center m-4' style='border: 2px solid white; padding: 30px; color: white;'><h4>Porcentagem de Luminosidade: </h4>");
client.print("<h6>");
client.print(LDR_value_porcentagem);
client.print("</h6></div><div class='m-4'>");
client.print("<div class='d-flex flex-wrap justify-content-around align-items-center'>");
client.print("<a class='btn btn-lg btn-success m-2' title='Ligar Luz' href='/Ligar'>Ligar Luz</a>");
client.print("<a class='btn btn-lg btn-danger m-2' title='Desligar Luz' href='/Desligar'>Desligar Luz</a>");
client.print("</div><div class='d-flex flex-wrap justify-content-around align-items-center'>");
if(dutyCycle != 255){
  client.print("<a class='btn btn-lg btn-primary m-2' title='Aumento em 10%' href='/Aumentar'>Aumentar Luminosidade</a>");
} else {
  client.print("<a class='btn btn-lg btn-primary m-2' title='Aumento em 10%' href='/Aumentar' disabled>Aumentar Luminosidade</a>");
}
if(dutyCycle != 0){
  client.print("<a class='btn btn-lg btn-secondary m-2' title='Reduz em 10%' href='/Reduzir'>Reduzir Luminosidade</a>");
} else {
  client.print("<a class='btn btn-lg btn-secondary m-2' title='Reduz em 10%' href='/Reduzir' disabled>Reduzir Luminosidade</a>");
}
client.print("</div></div></div></div></body></html>");
```

Fonte: Autoria própria

A partir desse código, conseguimos a seguinte página como resultado 6

Figura 6: Página HTML Resultante



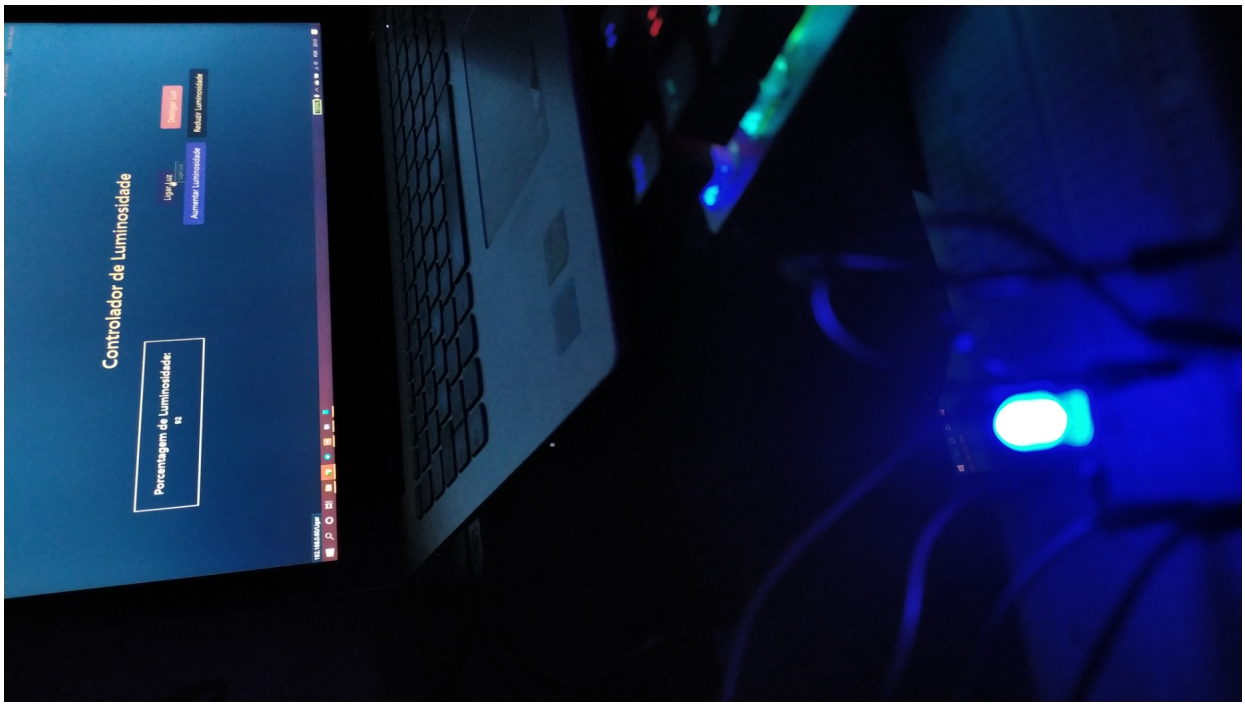
Fonte: Autoria própria

Outro ponto ah abordar são as funcionalidades, nesse pequeno projeto haviam quatro possibilidades de resultados: Ligar, Desligar, Aumentar Luminosidade, Reduzir Luminosidade. Como, já havia comentado anteriormente na secção de wi-fi, o esp 32 recebe esses comandos através de requisições GET a partir da URL do respectivo site, e o código que tratava essas requisições pode ser visto abaixo:

```
if (currentLine.endsWith("GET /Ligar")) {
  dutyCycle = 255;
  ledcWrite(ledChannel, dutyCycle); // GET /Ligar a luz é Ligada
}
if (currentLine.endsWith("GET /Desligar")) {
  dutyCycle = 0;
  ledcWrite(ledChannel, dutyCycle); // GET /Desligar a luz é Desligada
}
if (currentLine.endsWith("GET /Aumentar")) {
  if(!dutyCycle != 255) {
      dutyCycle = dutyCycle + 25.5;
      ledcWrite(ledChannel, dutyCycle); // GET /Aumentar a luminosidade é aumentada em 10%
   };
}
if (currentLine.endsWith("GET /Reduzir")) {
  if(dutyCycle != 0) {
      dutyCycle = dutyCycle - 25.5;
      ledcWrite(ledChannel, dutyCycle); // GET /Reduzir a luminosidade é reduzida em 10%
   };
}
```

Começando pelo botão de ligar, ao clicarmos nele o led acendeu perfeitamente como pode se ver na figura 7, chegando a uma porcentagem de 92. O que já é bastante satisfatório.

Figura 7: Resultado botão Ligar Luz



Fonte: Autoria própria

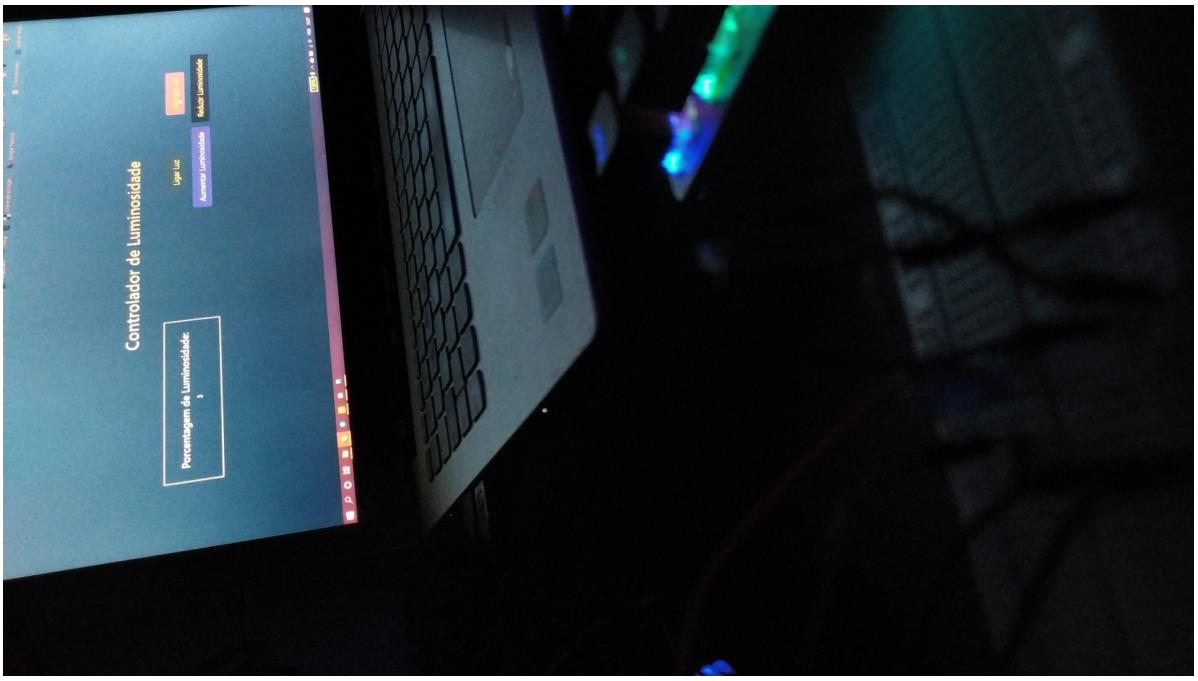
Passando para o botão de reduzir luminosidade, ao irmos clicando nele, como previsto o led foi perdendo intensidade, ou seja, luminosidade. Visualizando a figura 8 podemos ver uma queda na porcentagem, indo de 92 para 40. O que já é bastante satisfatório.

Figura 8: Resultado botão Reduzir Luminosidade

Fonte: Autoria própria

Agora para o botão de desligar, ao ser clicado como previsto o led foi é desligado. Visualizando a figura 9 podemos ver a porcentagem chegando a quase 0, como temos a presença do teclado que é luminoso, acaba gerando interferência no sensor. Mesmo assim, o resultado obtido é satisfatório, pois o led apagou e a pagina mostra ao usuário que não ah mais iluminação.

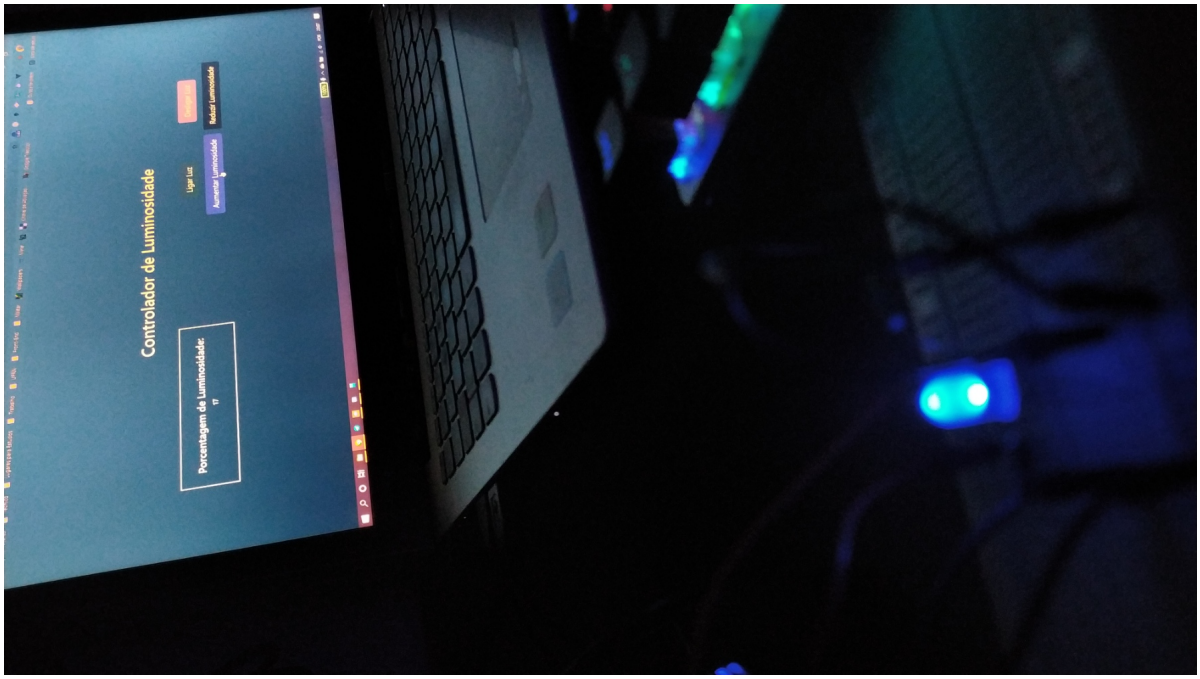
Figura 9: Resultado botão Desligar Luz



Fonte: Autoria própria

Por ultimo, temos o botão de aumentar luminosidade. Ao clicarmos nele aconteceu o oposto do botão de reduzir, a intensidade da luz aumentou como pode ser visto na figura 10. Demonstrando assim, o funcionamento esperado e conseguindo o resultado satisfatório.

Figura 10: Resultado botão Aumentar Luminosidade



Fonte: Autoria própria

Portanto, podemos destacar que o controlador funcionou como esperado pelo projeto e projetista. Para uma melhor visualização do funcionamento como um todo, um video de demonstração das funcionalidades desse projeto pode ser acessado através do link: https://drive.google.com/file/d/1HpHbiMZCBkXhTUZYlrckNppTH5AH8KpS/view?usp=sharing

4 CONCLUSÃO

Dado os ensinamentos em sala de aula e os exemplos encontrados na IDE do Arduíno, conseguir implementar o controlador de luminosidade com as funcionalidades esperadas. Podendo, não só no futuro gerar outros funcionalidades para esse controlador, como também, poder usar outros periféricos disponibilizados.

Referências

1 ROVAI, Marcelo José. “IOT feito fácil”: Brincando com o ESP32 no Arduino IDE. [S. l.], 2017. Disponível em: https://mjrobot.org/2017/09/26/iot-feito-facil-brincando-com-o-esp32-no-arduino-ide/. Acesso em: 29 mar. 2021.

2 DATASHEET ESP-Wroom-32. [S. l.], 2019. Disponível em: https://pdf1.alldatasheet.com/datasheet-pdf/view/1179101/ESPRESSIF/ESP-WROOM-32.html. Acesso em: 29 mar. 2021.