AmbiTemp

Pedro da Costa Cason, Pedro Henrique de Almeida Costa, Wilian França Costa

1Universidade Presbiteriana Mackenzie (UPM)  
Rua da Consolação, 930 Consolação, São Paulo - SP, 01302-907 – Brazil

pedrocason@outlook.com, pedro.costa08@hotmail.com

**Abstract.** This article describes an Arduino Project, “AmbiTemp” that has an alarm that will be triggered by a temperature sensor that can be read using the web.

**Resumo.** Este artigo descreve o projeto, “AmbiTemp”, utilizando Arduino que tem como objetivo reproduzir um alarme baseado em sensor de temperaturas, que será lido pelos usuários via web.

# Introdução

No AmbiTemp, será simulada uma situação de controle de temperaturas em ambientes. Em Data C enters, por exemplo, os equipamentos precisam de temperaturas controladas, normalmente abaixo de 20ºC. Se ocorrer uma falha no sistema de ventilação, a temperatura geralmente sobe, e pode ocasionar em perda de equipamentos, e às vezes, combustão.

No projeto, será utilizado um sensor de temperatura juntamente com um atuador “Buzzer”, que é um tipo de alarme. A temperatura será monitorada o tempo todo, e caso ultrapasse a temperatura indicada no código, o alarme dispara, junto com um LED vermelho. O alarme desarma quando a temperatura volta ao normal indicado no código.

O monitoramento poderá ser visualizado pelos usuários em aplicativos que apresentam dashboards de uso MQTT, como o “MQTT Dashboard”, disponível para iOS e Android, na comunicação realizada pelo microcontrolador, utilizando o padrão MQTT. Com os dados sendo enviados, o usuário saberá quando a temperatura estiver acima do limite e quando o alarme disparar.

# Materiais e Métodos

**2.1 - Os materiais que foram utilizados para a concepção do projeto são:**



Figura 1. Placa WeMos D1 ESP8266 - Disponível em : https://www.autocorerobotica.com.br/placa-de-desenvolvimento-esp8266-wemos-d1-r2

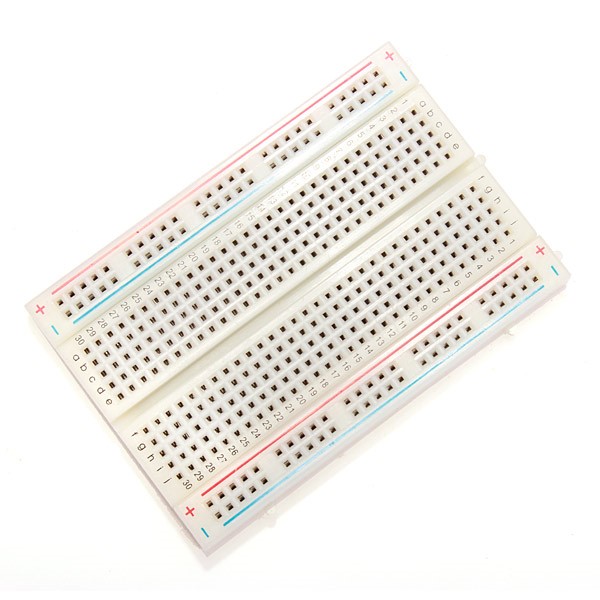


Figura 2. Protoboard 400 Pontos - Disponível em: https://www.filipeflop.com/produto/protoboard-400-pontos/

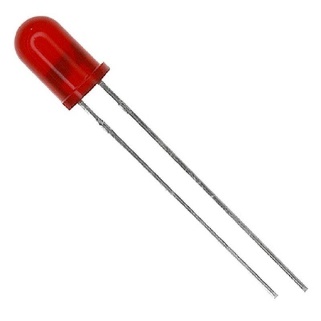


Figura 3. LED Vermelho 5mm difuso – Disponível em: https://www.filipeflop.com/produto/led-difuso-5mm-vermelho-x10-unidades/



Figura 4 – Resistor 150R 1/4W – Disponível em: https://www.usinainfo.com.br/resistores/resistor-150r-14w-kit-com-10-unidades-2991.html



Figura 5 - Sensor de Temperatura LM35DZ – Disponível em: https://www.filipeflop.com/produto/sensor-de-temperatura-lm35dz/



Figura 6 - Atuador Buzzer Ativo 5V – Autor: OLIVEIRA, Euler. Disponível em: https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-buzzer-5v-ativo/

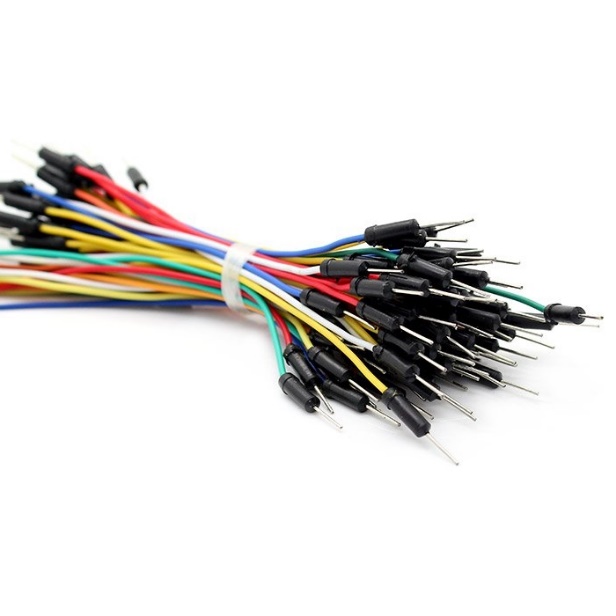


Figura 7 - Jumpers Macho/Macho – Disponível em: https://www.filipeflop.com/produto/kit-jumpers-macho-macho-x65-unidades/



Figura 8 - Fonte de Alimentação 9v P4 – Disponível: https://produto.mercadolivre.com.br/MLB-1028541728-fonte-de-alimentaco-carregador-9v-1a-plug-p4-arduino-aplic-\_JM?quantity=1

**2.2 – Métodos**

**2.2.1 – Código utilizado na IDE Arduino:**

**#include <PubSubClient.h>**

**#include <ESP8266WiFi.h>**

**#define BUZZER\_PIN D7**

**#define LED\_VERMELHO D5**

**const char\* ssid = "CASA";**

**const char\* password = "empresa2010";**

**const char\* mqtt\_server = "mqtt.eclipse.org";**

**const char\* mqttUser = "ambi";**

**const char\* mqttPassword = "5276";**

**const int mqttPort = 1883;**

**char pub\_str[100];**

**float gettemp();**

**WiFiClient espClient;**

**PubSubClient client(espClient);**

**void setup\_wifi()**

**{**

**delay(10);**

**Serial.print("Conectando na rede ");**

**Serial.println(ssid);**

**WiFi.begin(ssid, password);**

**while (WiFi.status() != WL\_CONNECTED){**

**delay(500);**

**Serial.print("-");**

**}**

**Serial.println();**

**Serial.println("WiFi Conectado");**

**Serial.println("IP ");**

**Serial.println();**

**Serial.println(WiFi.localIP());**

**}**

**void callback(char\* topic, byte\* payload, unsigned int length){**

**Serial.print("Mensagem recebida : ");**

**Serial.print(topic);**

**Serial.print(" : ");**

**pinMode(LED\_VERMELHO, OUTPUT);**

**for (int i = 0; i < length; i++){**

**Serial.println((char)payload[i]);**

**}**

**pinMode(LED\_VERMELHO, OUTPUT);**

**pinMode(BUZZER\_PIN, OUTPUT);**

**float val = analogRead(A0);**

**float c = val \* 285.0 / 1024.0;**

**if ((char)payload[0] == 'o' && (char)payload[1] == 'n'){**

**digitalWrite(LED\_VERMELHO, HIGH);**

**}**

**else if ((char)payload[0] == 'o' && (char)payload[1] == 'f' && (char)payload[2] == 'f') {**

**digitalWrite(LED\_VERMELHO, LOW);**

**}**

**}**

**void reconnect(){**

**while(!client.connected()){**

**Serial.println("Tentando conexão com o MQTT");**

**if(client.connect("...")){**

**Serial.println("Conectado");**

**client.publish("ambitemp\_temperatura","Conectado!");**

**client.subscribe("ambitemp\_led");**

**Serial.print("Inscrito!");**

**}**

**else{**

**Serial.print("Falha, rc = ");**

**Serial.print(client.state());**

**Serial.println("Nova tentativa em 5 segundos ");**

**delay(5000);**

**}**

**}**

**void setup(){**

**pinMode(LED\_VERMELHO, OUTPUT);**

**pinMode(BUZZER\_PIN, OUTPUT);**

**digitalWrite(BUZZER\_PIN, LOW);**

**Serial.begin(115200);**

**setup\_wifi();**

**client.setServer(mqtt\_server, 1883);**

**client.setCallback(callback);**

**reconnect();**

**}**

**void loop()**

**{**

**if(!client.connected()){**

**reconnect();**

**Serial.print("Desconectado");**

**}**

**float tmp = gettemp();**

**dtostrf(tmp,2,2,pub\_str);**

**Serial.print(pub\_str);**

**Serial.println(tmp);**

**client.publish("ambitemp\_temperatura",pub\_str);**

**delay(1000);**

**client.loop();**

**}**

**float gettemp() {**

**float val = analogRead(A0);**

**float c = val \* 285.0 / 1024.0;**

**if (c > 27) {**

**digitalWrite(LED\_VERMELHO, HIGH);**

**digitalWrite(BUZZER\_PIN, HIGH);**

**}**

**else{**

**digitalWrite(LED\_VERMELHO, LOW);**

**digitalWrite(BUZZER\_PIN, LOW);**

**}**

**Serial.print("Temperatura = ");**

**return c;**

**}**

**2.2.2 – Protocolo utilizado**

No projeto, foi utilizado o protocolo de comunicações MQTT (MQ Telemetry Transport). Para evitar cobranças, utilizamos o Broker da Eclipse, com as seguintes configurações:

* Servidor MQTT: "mqtt.eclipse.org";
* Usuário: "ambi";
* Senha: "5276";
* Porta de Comunicação: 1883;
* Tópico do mqtt: ambitemp\_temperatura

Com estes dados, foi possível conectar o circuito no mqtt. Assim, com a ajuda de um aplicativo (usamos o “MQTT Dash”), podemos ter a visualização das temperaturas diretamente no smartphone:

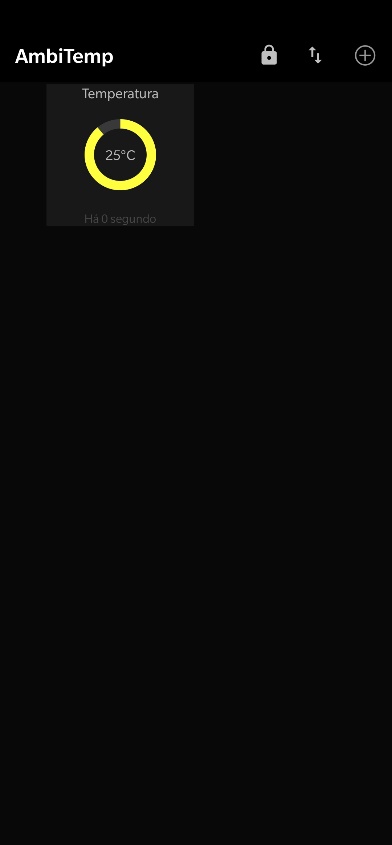


Figura 9 – Visualização da temperatura via MQTT no smartphone. Fonte própria

**2.2.3 – Fluxograma de funcionamento**

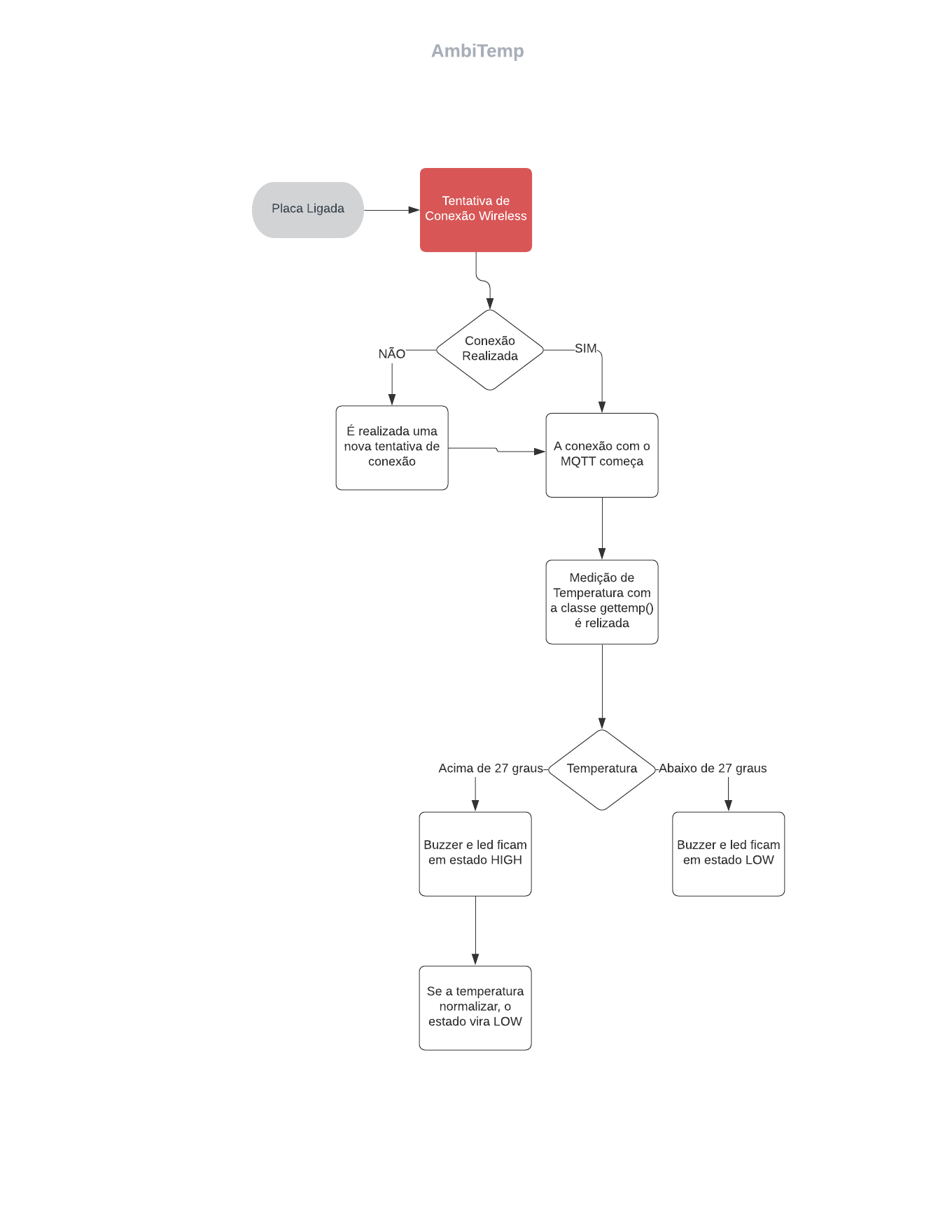
****

Figura 9 - Fluxograma realizado no "LucidChart"

# 2.2.4. Estrutura básica utilizando o TinkerCad

Utilizando o TinkerCad, foi realizado um projeto com a estrutura básica que será utilizada no projeto físico. No projeto oficial estamos utilizando uma placa WeMos D1, que não está disponível no TinkerCad, mas o esquema é exatamente o mesmo:

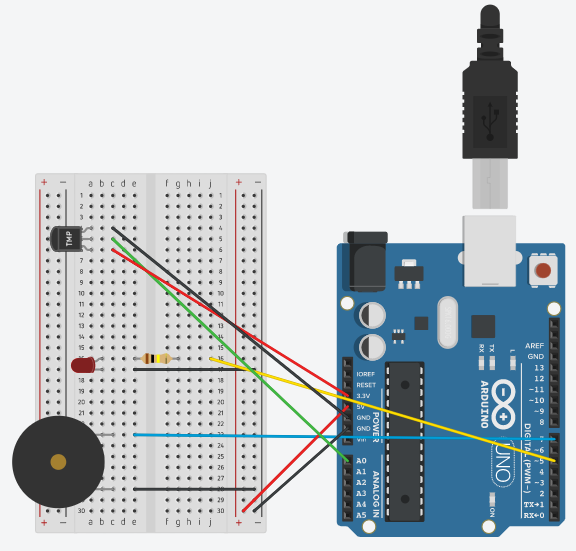


Figura 10 - Projeto TinkerCad: Spectacular Trug-Gaaris. Disponível em: https://www.tinkercad.com/things/5lzZMgvcVvZ

1. **Resultados**

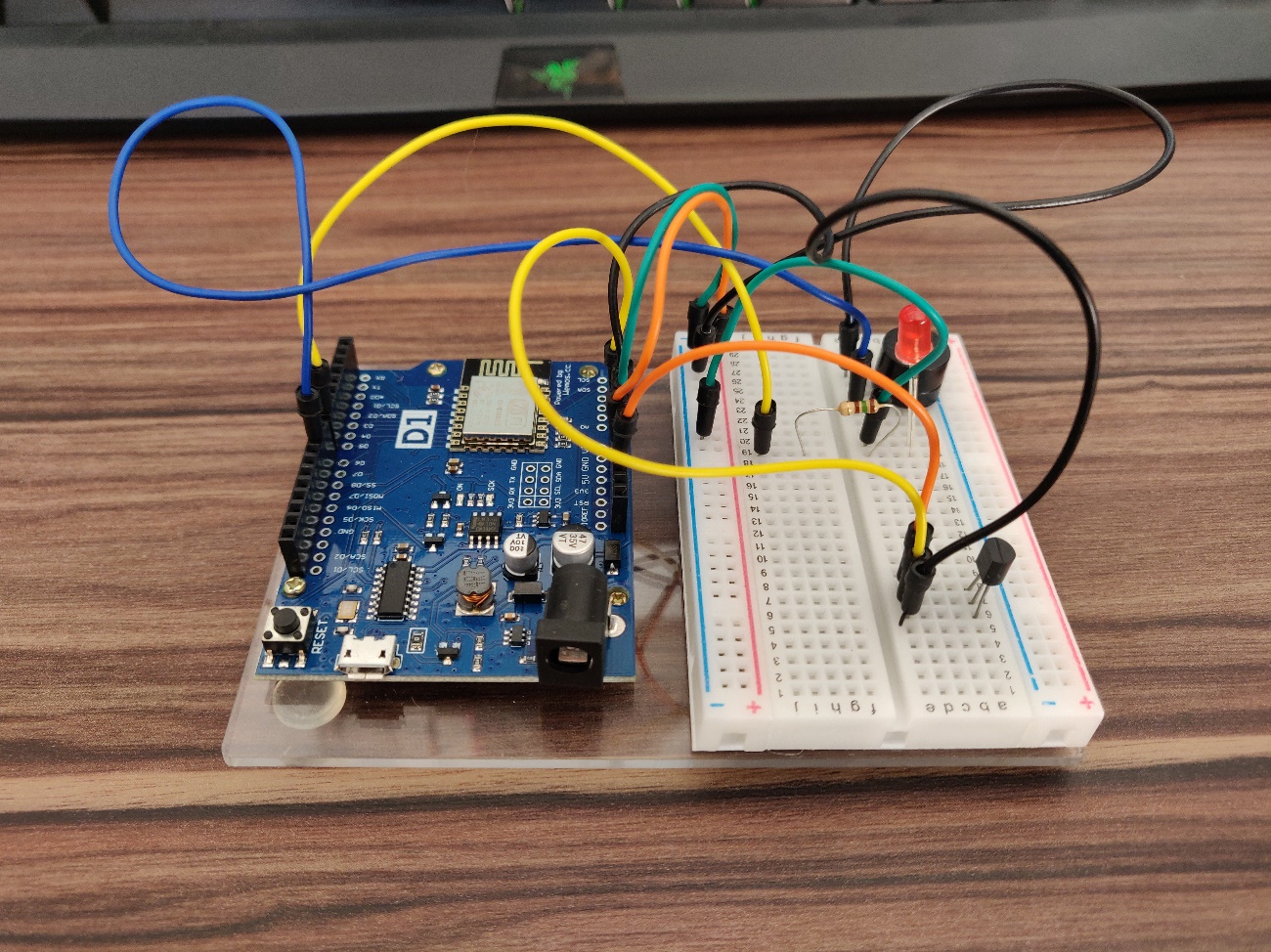
****

Figura 7 - Imagem do circuito finalizado. Fonte propria

* 1. **– Funcionamento**
  2. **- Principais problemas enfrentados**

Houve alguns problemas no início do desenvolvimento do projeto. Entre eles, após o recebimento dos produtos, fizemos as ligações de maneira errônea, e os componentes soltaram um cheiro de “queimado”. Achamos que os componentes haviam sido perdidos. Mas, após realizarmos as ligações de forma correta, os componentes funcionaram sem problemas e sem o cheiro.

Na programação do projeto, também tivemos problemas quanto a apresentação da temperatura do sensor, que apresentava temperaturas fora das faixas aceitáveis (a temperatura inicial ficava na casa dos 40ºC). Mas após uma pesquisa sobre as fórmulas para a apresentação correta das temperaturas, o problema foi resolvido.

Outro problema enfrentado foi a programação do payload para em caso de “On”, manter o sistema de alarme e led funcionando. E em caso de “Off”, desligar todo o sistema. Tentamos diversas vezes programar de forma correta esta função mas não obtivemos sucesso. Em várias das vezes, no payload em “On”, o sistema ficava com o Buzzer e o Led ligados indefinidamente, mesmo que a temperatura estivesse abaixo dos 27ºC definidos no software.

# Conclusões

* 1. **– Objetivos**

O objetivo proposto foi alcançado. A ideia inicial era que fosse possível criar um sistema de controle de temperatura, onde a temperatura pudesse ser observada pelo smartphone, sem a obrigação de olhar no serial da placa.

* 1. **– Vantagens e Desvantagens**
     1. **- Vantagens**

Este projeto tem como vantagem a possibilidade de controlar e avaliar a temperatura de um ambiente, sem a necessidade de estar fisicamente no mesmo. Assim, poderia ser implementado em um Data Center, ou uma casa com projeto de automação residencial.

* + 1. **– Desvantagens**

Infelizmente, não foi possível colocar no sistema, um botão no smartphone para controlar o funcionamento do LED e do Buzzer. Sendo assim, o sistema não pode ser desativado (apenas se tirar da fonte de energia). E sempre que a temperatura passar do programado, o sistema irá funcionar.

* 1. **– O que poderia ser feito para melhorar o projeto**

Para a melhoria no funcionamento do projeto, acreditamos que seria realmente interessante a implementação do tópico MQTT para desarmar o sistema, assim ficando a cargo do usuário, decidir se gostaria de manter o sistema ligado ou não.

Assim como deixar o sistema mais sofisticado, como a implementação de um sistema de mensageria. Caso o sistema ultrapassasse os 27ºC, além do Buzzer apitando, o usuário receberia um email de alerta, com um aviso de que a temperatura do ambiente X ultrapassou o limite recomendado.

# 5. Referências

BAUERMEISTER, Giovanni. Primeiros passos com Arduino. **FILIPEFLOP**, 2018. Disponível em: <https://www.filipeflop.com/blog/primeiros-passos-com-arduino/>. Acesso em: 06, abril de 2020.

ELECTRONICS, Mouser. Sensores e Atuadores IoT. **EMBARCADOS,** 2020.

Disponível em: < https://www.embarcados.com.br/sensores-e-atuadores-iot/>.

Acesso em: 06, abril de 2020.

THOMSEN, Adilson. Tutorial Módulo Wireless ESP8266 com Arduino. **FILIPEFLOP**, 2015.

Disponível em: < https://www.filipeflop.com/blog/esp8266-arduino-tutorial/>.

Acesso em: 07, abril de 2020.

OLIVEIRA, Euler. Sensor de Temperatura TMP36. **MasterWalker BLOG**.

Disponível em: <https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-sensor-de-temperatura-tmp36/>

Acesso em: 07, abril de 2020

MOTA, Allan. USANDO O BUZZER COM ARDUINO - TRANSDUTOR PIEZO ELÉTRICO. **Vida de Silicio**, 2017.

Disponível em: <https://portal.vidadesilicio.com.br/usando-o-buzzer-com-arduino-transdutor-piezo-eletrico/>

Acesso em: 07, abril de 2020

CIA, Arduino. Enviando mensagens MQTT com módulo ESP32. **Arduino e Cia,** 2019.

Disponível em: <https://www.arduinoecia.com.br/enviando-mensagens-mqtt-modulo-esp32-wifi/>

Acesso em: 18, junho de 2020

SOUZA, Fábio. Use o MQTT Dash para controlar uma lâmpada remotamente. **Embarcados**, 2018.

Disponível em: <https://www.embarcados.com.br/mqtt-dash/>

Acesso em: 18, junho de 2020

SOUZA, Fábio. Exibindo valores de temperatura e umidade no app MQTT Dash com o sensor DHT11. **Embarcados**, 2018.

Disponível em: <https://www.embarcados.com.br/temperatura-e-umidade-no-mqtt-dash-dht11/>

Acesso em: 19, junho de 2020.

Padeath. Remote Temperature Monitoring Using MQTT and ESP8266 Modules. **Instructables**.

Disponível em: <https://www.instructables.com/id/Remote-Temperature-Monitoring-Using-MQTT-and-ESP82/>

Acesso em: 19, junho de 2020

BERTOLETI, Pedro. Controle e Monitoramento IoT com NodeMCU e MQTT. **FILIPEFLOP**, 2016

Disponível em: <https://www.filipeflop.com/blog/controle-monitoramento-iot-nodemcu-e-mqtt/>

Acesso em: 19, junho de 2020