Bruno Bergamini Do Nascimento

Pedro Henrique Moreira

Umidificador Automático

Relatório:

Primeiramente tivemos como ideia construir um umidificador em pequena escala, voltado para locais menores e fechados como bares, restaurantes, shoppings, lojas etc. mantendo assim as condições de permanência, aceitáveis.

Pensamos em um sistema que liga um umidificador de ar, baseado nas condições de umidade atuais, caso o sensor identifique umidade menor que 30%, o que é considerado grave, o umidificador é ligado. Para fazer o projeto utilizamos um ESP32 como gerenciador do projeto, um sensor de umidade, um relé e construímos um umidificador de ar utilizando uma caixa de plástico e um cooler. O projeto também possuí conexão Wi-fi, comunicando para o usuário, via Telegram, as ações que o sistema irá fazer. As conexões foram feitas com jumpers fêmeas, e o cooler foi alimentado por uma bateria de 9v.

O primeiro problema que encontramos no desenvolvimento deste trabalho foi o acionamento da porta do relé, tendo em vista que alguns sensores e equipamentos utilizam a notação 0 para ligado e 1 para desligado, ficamos bastante tempo procurando a solução e decidimos pedir a ajuda ao professor Afonso Miguel. Após esse processo conseguimos juntar o sensor e o relé, fazendo o acionamento via script na IDE Thonny, utilizando uma condição que checava se a umidade do ar estava abaixo de 30%. Manuseamos um secador de cabelo para secar o ar e assim proceder os testes.

O segundo problema foi achar um jeito de ligar o cooler ao relé, o Esp32 só trabalha com 5v e o cooler precisava de 12v, assim não podíamos ligar o cooler direto na tomada pois ele seria incinerado. Utilizamos uma bateria de 9v para alimentar o cooler. No inicio não deu certo, pois ou o cooler ficava ligado o tempo todo, ou sequer ligava, pedimos novamente ajuda ao professor e o mesmo revelou a solução do problema, o problema é que estava faltando um pequeno jumper no relé, e ausência do mesmo barrava a corrente da bateria, passando ao relé somente a energia de 5v do Esp32 que era insuficiente.

Depois disso fizemos a integração com o Telegrambot (BotFather), enviando uma mensagem para o usuário quando o cooler fosse ligado, pra fazer isso pegamos um código pronto da internet e o reconfiguramos para as nossas necessidades. Nesse ponto tivemos que trocar a ide que estávamos utilizando que no caso era o Thonny e passamos a usar a Arduino IDE, pois todo o processo requeria a instalação e importação de um pacote JSON e a própria biblioteca Telegrambot.

Com o acionamento do cooler e a integração com o Telegram pronta, passamos a fazer o umidificador. Utilizamos como fonte um vídeo do Manual do Mundo que ensinava como fazer um umidificador caseiro, utilizando um pote do tipo Tupperware, panos, água e o cooler.

Link do vídeo:

https://www.youtube.com/watch?v=fu3niqZ-nD4&ab_channel=PedroMoreira

Código utilizado:

```
#if defined(ESP8266)
#include <ESP8266WiFi.h>
#else
#include <WiFi.h>
#endif
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <SimpleDHT.h>

//Pino onde está o Relê
#define RELAY_PIN 13
//Pino onde está o DHT22
#define DHT_PIN 15

//Intervalo entre as checagens de novas mensagens
#define INTERVAL 1000

//Token do seu bot. Troque pela que o BotFather te mostrar
#define BOT_TOKEN "1474768750:AAEbnSed0ZQJYqpLa91YWGxDjDnnKuOkXnM"

//Troque pelo ssid e senha da sua rede WiFi
#define SSID "XXXXXXXX"
```

```
#define PASSWORD "XXXXXXXX"

//Comandos aceitos
const String CLIMATE = "clima";
const String STATS = "status";
const String START = "/start";

//Objeto que realiza a leitura da temperatura e umidade
SimpleDHT11 dht;

//Estado do relê
int relayStatus = HIGH;

//Cliente para conexões seguras
WiFiClientSecure client;
//Objeto com os métodos para comunicarmos pelo Telegram
UniversalTelegramBot bot(BOT_TOKEN, client);
//Tempo em que foi feita a última checagem
uint32_t lastCheckTime = 0;

//Quantidade de usuários que podem interagir com o bot
#define SENDER_ID_COUNT 2
//Ids dos usuários que podem interagir com o bot.
//É possível verificar seu id pelo monitor serial ao enviar uma mensagem para o bot
String validSenderIds[SENDER_ID_COUNT] = {"XXXXXXXXX"};

void setup()
{
  Serial.begin(115200);
```

```
  //Inicializa o WiFi e se conecta à rede
  setupWiFi();

  //Coloca o pino do relê como saída e enviamos o estado atual
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, relayStatus);
}

void setupWiFi()
{
  Serial.print("Connecting to SSID: ");
  Serial.println(SSID);

  //Inicia em modo station e se conecta à rede WiFi
  WiFi.mode(WIFI_STA);
  WiFi.begin(SSID, PASSWORD);

  //Enquanto não estiver conectado à rede
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(500);
  }

  //Se chegou aqui está conectado
  Serial.println();
  Serial.println("Connected");
}

void handleNewMessages(int numNewMessages)
```

```
{
  for (int i=0; i<numNewMessages; i++) //para cada mensagem nova
  {
    String chatId = String(bot.messages[i].chat_id); //id do chat
    String senderId = String(bot.messages[i].from_id); //id do contato

    Serial.println("senderId: " + senderId); //mostra no monitor serial o id de quem mandou a mensagem

    boolean validSender = validateSender(senderId); //verifica se é o id de um remetente da lista de remetentes válidos

    if(!validSender) //se não for um remetente válido
    {
      bot.sendMessage(chatId, "Desculpe mas você não tem permissão", "HTML"); //envia mensagem que não possui permissão e retorna sem fazer mais nada
      continue; //continua para a próxima iteração do for (vai para próxima mensgem, não executa o código abaixo)
    }

    String text = bot.messages[i].text; //texto que chegou

    if (text.equalsIgnoreCase(START))
    {
      handleStart(chatId, bot.messages[i].from_name); //mostra as opções
    }
    else if(text.equalsIgnoreCase(CLIMATE))
    {
      handleClimate(chatId); //envia mensagem com a temperatura e umidade
    }
    else if (text.equalsIgnoreCase(STATS))
```

```
    {
      handleStatus(chatId); //envia mensagem com o estado do relê, temperatura e umidade
    }
    else
    {
      handleNotFound(chatId); //mostra mensagem que a opção não é válida e mostra as opções
    }
  }//for
}

boolean validateSender(String senderId)
{
  //Para cada id de usuário que pode interagir com este bot
  for(int i=0; i<SENDER_ID_COUNT; i++)
  {
    //Se o id do remetente faz parte do array retornamos que é válido
    if(senderId == validSenderIds[i])
    {
      return true;
    }
  }

  //Se chegou aqui significa que verificou todos os ids e não encontrou no array
  return false;
}

void handleStart(String chatId, String fromName)
{
  //Mostra Olá e o nome do contato seguido das mensagens válidas
```

```
  String message = "<b>Olá " + fromName + ".</b>\n";
  message += getCommands();
  bot.sendMessage(chatId, message, "HTML");
}

String getCommands()
{
  //String com a lista de mensagens que são válidas e explicação sobre o que faz
  String message = "Os comandos disponíveis são:\n";
  message += "<b>" + CLIMATE + "</b>: Para verificar o clima\n";
  message += "<b>" + STATS + "</b>: Para verificar o estado do umidificador, a temperatura e umidade";
  return message;
}

void handleClimate(String chatId)
{
  //Envia mensagem com o valor da temperatura e da umidade
  bot.sendMessage(chatId, getClimateMessage(), "");
}

String getClimateMessage()
{
  //Faz a leitura da temperatura e da umidade
  float temperature, humidity;
  int status = dht.read2(DHT_PIN, &temperature, &humidity, NULL);

  //Se foi bem sucedido
  if (status == SimpleDHTErrSuccess)
  {
    //Retorna uma string com os valores
```

```
    String message = "";
    message += "A temperatura é de " + String(temperature)+ " °C e ";
    message += "a umidade é de " + String(humidity) + "%";
    return message;
  }

  //Se não foi bem sucedido retorna um mensagem de erro
  return "Erro ao ler temperatura e umidade";
}

void handleStatus(String chatId)
{
  String message = "";

  //Verifica se o relê está ligado ou desligado e gera a mensagem de acordo
  if(relayStatus == LOW) //A lógica do nosso relê é invertida
  {
    message += "O umidificador está ligado\n";
  }
  else
  {
    message += "O umidificador está desligado\n";
  }

  //Adiciona à mensagem o valor da temperatura e umidade
  message += getClimateMessage();

  //Envia a mensagem para o contato
  bot.sendMessage(chatId, message, "");
}
```

```
void handleNotFound(String chatId)
{
  //Envia mensagem dizendo que o comando não foi encontrado e mostra opções de comando válidos
  String message = "Comando não encontrado\n";
  message += getCommands();
  bot.sendMessage(chatId, message, "HTML");
}



boolean jaPrintouAceso = false;
boolean jaPrintouDesligado = false;
void loop()
{
  //Tempo agora desde o boot
  uint32_t now = millis();

  //Se o tempo passado desde a última checagem for maior que o intervalo determinado
  if (now - lastCheckTime > INTERVAL)
  {
    //Coloca o tempo de útlima checagem como agora e checa por mensagens
    lastCheckTime = now;
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    handleNewMessages(numNewMessages);
  }
    delay(2000);
    //Faz a leitura da temperatura e da umidade
    float humidity, temp;
    int status = dht.read2(DHT_PIN, &temp, &humidity, NULL);
```

```
    //Se foi bem sucedido
    if (status == SimpleDHTErrSuccess)
    {
     if (humidity < 30){
      //Retorna uma string com os valores
      //Liga o relê e envia mensagem confirmando a operação
      relayStatus = LOW; //A lógica do nosso relê é invertida
      digitalWrite(RELAY_PIN, relayStatus);
        if(jaPrintouAceso == false){
          bot.sendMessage("1060025996", "A umidade está muito baixa, seu umidificador foi ligado.", "HTML");
          jaPrintouAceso = true;
          jaPrintouDesligado = false;
        }
   }else{
      relayStatus = HIGH;
      digitalWrite(RELAY_PIN, relayStatus);
      if(jaPrintouDesligado == false){
        bot.sendMessage("1060025996", "A umidade está normal, seu umidificador foi desligado.", "HTML");
        jaPrintouDesligado = true;
        jaPrintouAceso = false;
    }
   }
 }
}
```