**1)TECNOLOGIAS ESCOLHIDAS**

**a) Descreva as caracterísicas da placa de desenvolvimento escolhida:**



O ESP32 é um microcontrolador de baixo custo e de baixa potência projetado para aplicações de Internet das Coisas (IoT). Algumas das características são:

**Wi-Fi e Bluetooth:** Possui conectividade Wi-Fi integrada e suporte para Bluetooth Classic e BLE (Bluetooth Low Energy).

**Processador dual-core:** Possui dois núcleos de processamento.

**Memória:** Possui 520KB de SRAM e 4MB de memória flash.

**Baixo consumo de energia:** Torna-o ideal para dispositivos alimentados por bateria.

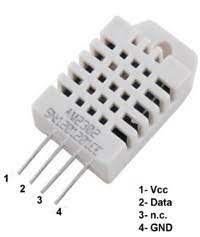
**Pinos de E/S:** Tem 34 pinos de entrada/saída (GPIO), que podem ser usados para controlar outros dispositivos ou receber entradas de sensores.

**Capacidade de processamento de sinais analógicos:** Possui um conversor analógico-digital (ADC) integrado, permitindo que ele processe sinais analógicos, como dados de sensores.

**Segurança:** Possui recursos de segurança, como criptografia e autenticação.

**Suporte a protocolos:** Suporta uma ampla variedade de protocolos de comunicação, como MQTT, HTTP e WebSocket.

**b) Descreva o sensor escolhido para captura das variáveis climáticas:**

****

O DHT22 é um sensor de umidade e temperatura digital que utiliza um único pino de comunicação para transmitir os dados medidos. Algumas das características são:

**Faixa de medição**: Pode medir a umidade relativa do ar de 0% a 100% com uma precisão de ±2% e a temperatura de -40°C a +80°C com uma precisão de ±0,5°C.

**Alta precisão:** É um sensor de alta precisão, o que o torna ideal para aplicações que exigem medições precisas de umidade e temperatura.

**Fácil de usar:** Fácil de usar, com um único pino de dados que transmite as leituras de umidade e temperatura em formato digital.

**Rápida resposta:** Tem uma resposta rápida e pode fornecer leituras atualizadas a cada dois segundos.

**Baixo consumo de energia:** Tem um baixo consumo de energia e pode ser usado em aplicações que requerem longa vida útil da bateria.

**Calibração interna:** Possui um circuito interno de calibração, o que significa que ele é capaz de autocalibrar-se em condições normais de operação.

**Baixo custo:** É um sensor de baixo custo.

**c) Lista dos componentes utilizados e valores:**

|  |  |  |
| --- | --- | --- |
| **NOME** | **QUANTIDADE** | **PREÇO** |
|  |  |  |
| **Placa ESP32** | **1** | **R$ 45,00** |
| **Sensor DHT22** | **1** | **R$ 35,00** |
| **LED Vermelha** | **1** | **R$ 0,35** |
| **LED Amarela** | **1** | **R$ 0,35** |
| **Resistor 100**Ω | **2** | **R$ 0,13** |
|  |  |  |
| **TOTAL** |  | **R$ 80,96** |

**Obs:** Os valores coletados foram feitos por base de uma média realizada entre os sites [www.mercadolivre.com](http://www.mercadolivre.com) e [www.shopee.com](http://www.shopee.com) na data de 22/04/2023.

\*Os resistores usados no design do circuito foram de 90Ω, entretanto, os resistores escolhidos foram os de 100Ω, visto que são comercialmente viaveis e também cumprem com os requisitos para dissipar a energia excedente e alimentar os leds de forma segura.

**2) CIRCUITO DESENVOLVIDO**

|  |  |
| --- | --- |
| **Temperatura <= 35ºC**  **Umidade <= 70%** |  |
| **Temperatura > 35ºC**  **Umidade <= 70%** |  |
| **Temperatura <= 35ºC**  **Umidade > 70%** |  |
| **Temperatura > 35ºC**  **Umidade > 70%** |  |

**3) CÓDIGO DESENVOLVIDO**

#include <DHTesp.h> //Inclusão da biblioteca DHTesp.h;

DHTesp dhtSensor; //Declara um objeto da classe DHTesp;

//Declaração dos valores dos pinos a qual os leds estão conectados;

int led1 = 18;

int led2 = 19;

//Declaração dos valores mínimos usados nas condicionais;

int medTemperatura = 35;

int medUmidade = 70;

void setup() {

  Serial.begin(115200);

  Serial.println("Inicio");

  //O sensor é do tipo DHT22 e esta conectado no pino 13 da placa;

  dhtSensor.setup(13, DHTesp::DHT22);

pinMode(led1, OUTPUT);   //O pinMode recebe o n° do pino e o tipo de saída;

  pinMode(led2, OUTPUT);

  Serial.println("Fim");

}

void loop() {

  //getTemperature e getHumidity são funções do DHTesp que retornam o valor;

  float temperatura = dhtSensor.getTemperature();

  float umidade = dhtSensor.getHumidity();

  Serial.println("Temperatura: " + String(temperatura) + "C");

  Serial.println("Umidade: " + String(umidade) + "%");

  Serial.println("--------------------------------------");

  //Se a temperatura colhida pelo sensor for maior que a declarada, a alimentação sera feita;

  if(temperatura > medTemperatura){

    digitalWrite(led1, HIGH); //Alimenta o pino led1;

  }

  else{

    digitalWrite(led1, LOW); //Não alimenta o pino led1;

  }

  //Se a umidade colhida pelo sensor for maior que a declarada, a alimentação sera feita;

  if(umidade > medUmidade){

    digitalWrite(led2, HIGH); //Alimenta o pino led2;

  }

  else{

    digitalWrite(led2, LOW); //Não alimenta o pino led2;

  }

  //Intervalo de 5000ms == 5s;

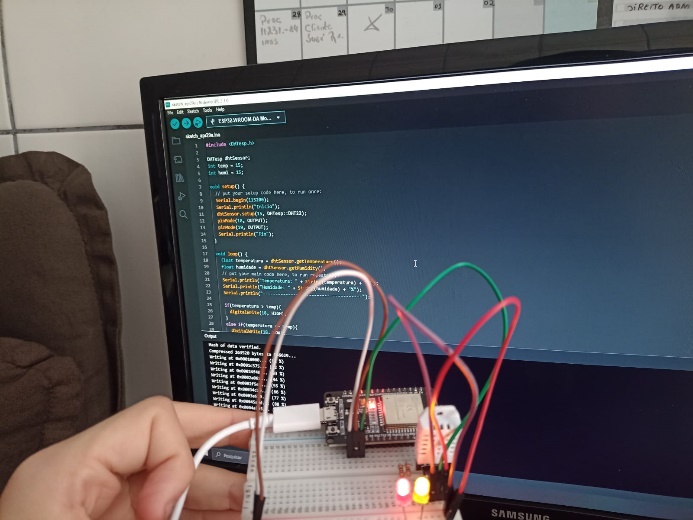
  delay(5000);

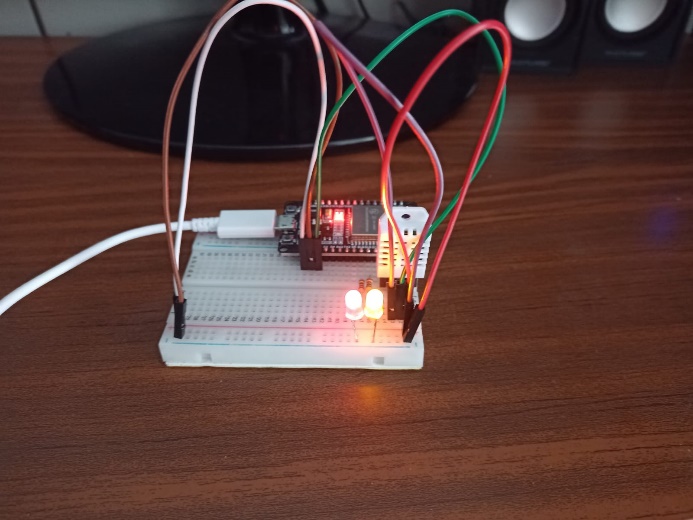
}

**4) EXTRA (APENAS PARA DEMONSTAR)**

Atividade feita de forma física com os itens listados no item 2, acrescentando apenas uma protoboard e jumpers para não utilizar solda; os 2 resistores usados são de 220Ω

Neste exemplo, reduzi as variáveis de temperatura e umidade para 15ºC e 15%, respectivamente.





Os atuadores são os dois leds, vermelho e amarelo, que representam temperatura e umidade, respectivamente. Como as condições de temperatura e umidade foram atingidas, os atuadores receberam alimentação.