UNIVERSIDADE DE ARARAQUARA

Projeto IoT Educativo de Controle de Estufa de Plantas no Wokwi

UNIVERSIDADE DE ARARAQUARA

Sumário

Lista de Figuras

1. Introdução ao Projeto IoT

Este projeto apresenta, de forma didática, o Monitoramento de uma Estufa de Plantas usando uma placa ESP32. O sistema realiza leituras de temperatura e umidade (em simulação, por meio de valores gerados no próprio código) e compara esses valores com limites pré-definidos. As informações são exibidas em um Display LCD 20x4, enquanto LEDs e um Buzzer são utilizados para sinalizar o estado do ambiente.

Internet das Coisas (IoT) é quando objetos do dia a dia (sensores, máquinas, aparelhos e dispositivos) se conectam a sistemas digitais para coletar dados, processar informações, tomar decisões e, quando necessário, acionar recursos (atuadores) ou se comunicar com outros sistemas.

A proposta é ensinar, passo a passo, como montar um sistema IoT simples, explicando a lógica utilizada e mostrando como iniciantes podem adaptar esse modelo para outras aplicações.

Neste projeto, temos uma “Mini-Estufa Inteligente”:

- O sistema trabalha com valores de temperatura e umidade (simulados), usando lógica equivalente ao que seria feito com um sensor DHT22 real;
- O ESP32 processa os dados e decide se está tudo dentro do esperado;
- O LCD 20x4 exibe as medições e o estado do sistema;
- O LED Verde indica condição normal;
- O LED Vermelho e o Buzzer disparam quando a temperatura estiver muito alta ou quando a umidade estiver muito alta (situação de alerta: “muito quente” ou “muito úmido”).

2. Desenvolvimento do Projeto Controle de Estufa de Plantas

Nesta seção, explicaremos de forma clara e didática o funcionamento do protótipo Controle de Estufa de Plantas.

O objetivo é permitir que iniciantes compreendam os conceitos de Internet das Coisas (IoT) aplicados no projeto e consigam replicá-lo com orientação, por meio da descrição simplificada dos componentes, das conexões básicas, da lógica de funcionamento e do passo a passo de execução em simulação (Wokwi).

2.1. Materiais Necessários para o Projeto

2.1.1. Componentes Eletrônicos

Neste projeto, foram utilizados os seguintes componentes para o desenvolvimento:

- 01 x ESP32 (placa de desenvolvimento)
- 01 x DHT22 (sensor de temperatura e umidade do ar)
- 01 x Display LCD 2004 com módulo I2C (4 linhas x 20 colunas)

- 01 x LED Vermelho
- 01 x LED Verde
- 02 x Resistores de 150 Ω
- 01 x Buzzer Piezoelétrico

2.1.2. Software e Ferramentas

Neste projeto, foram utilizadas as seguintes ferramentas e recursos de desenvolvimento:

- **Wokwi:** simulador online de circuitos e microcontroladores, utilizado para montar e testar o protótipo em ambiente virtual;
- **Simulação com ESP32:** execução e validação da lógica embarcada no ESP32, incluindo leitura (simulada) dos sensores e acionamento dos atuadores;
- **Bibliotecas utilizadas:** DHT sensor library for ESPx (para leitura/simulação do DHT22) e LiquidCrystal I2C (para controle do Display LCD via interface I2C).

2.2. Abertura do Projeto

Para a montagem do projeto você deve acessar o site Wokwi (https://wokwi.com/), clicar na placa ESP32 e depois clicar no Template da imagem ESP32 que esta na seção “Start Templates”.

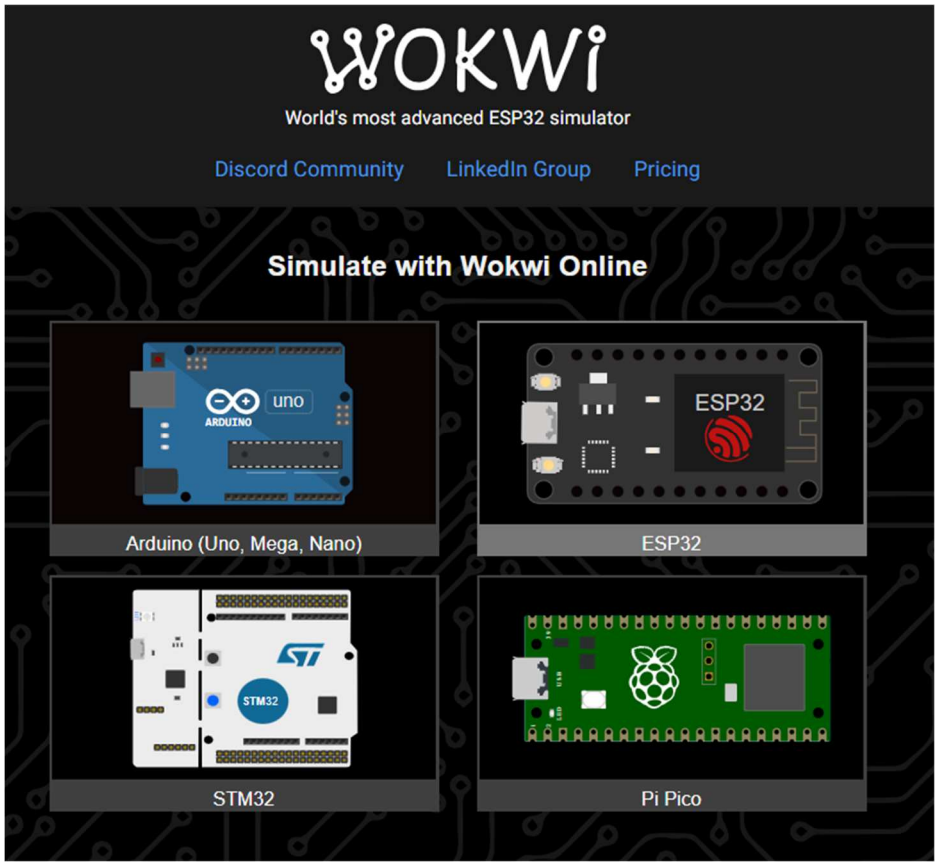

Figura 1 – Site Wokwi

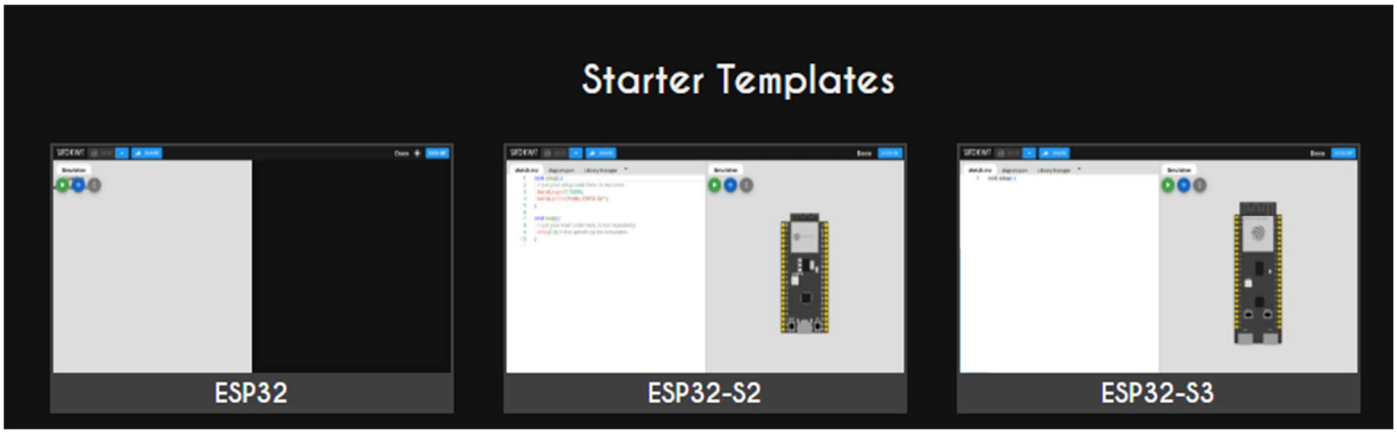

Figura 2 – Template ESP32

2.3. Montagem dos Componentes do Projeto

Para montar os componentes no simulador, clique no botão “+” para abrir a lista de opções disponíveis e adicione os itens necessários ao projeto. Caso deseje visualizar detalhes de algum componente, clique sobre ele e, em seguida, selecione o ícone de interrogação (?), onde serão exibidas as informações correspondentes.

Após inserir todos os componentes, prossiga com a montagem das conexões conforme o esquema indicado nas próximas etapas.

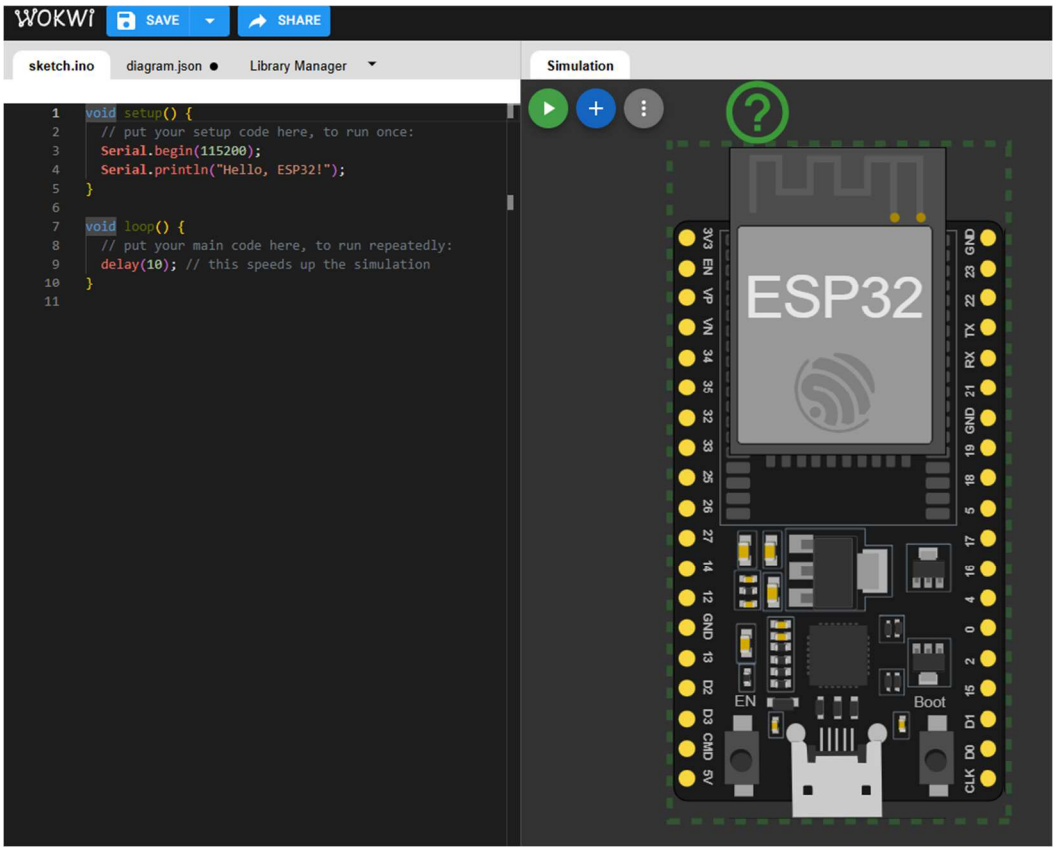

Figura 3 – Simulador Wokwi

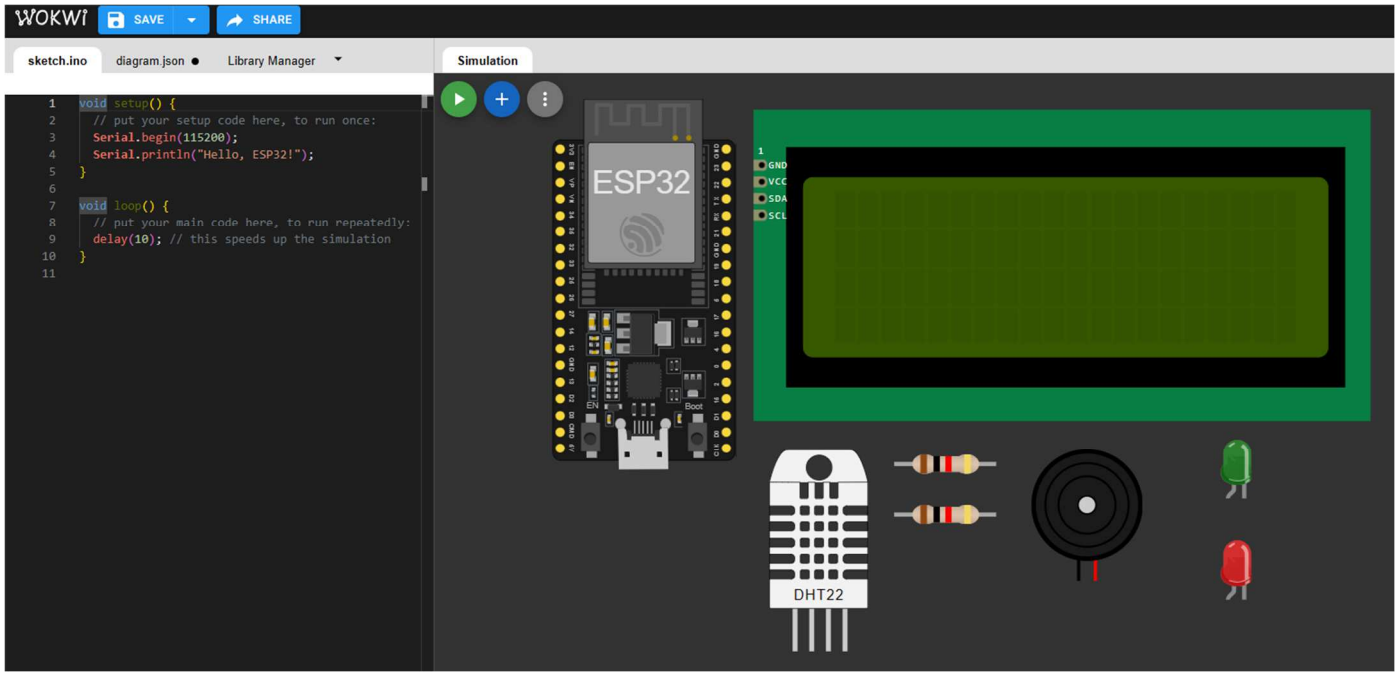

Figura 4 – Componentes para o Projeto

2.4. Ligações do Circuito do Projeto

Após adicionar todos os componentes no simulador e posicioná-los na área de montagem, realize as conexões conforme a figura/esquema do circuito do projeto.

Para ajustar o resistor, clique sobre o componente e altere o seu valor para 150 Ω.

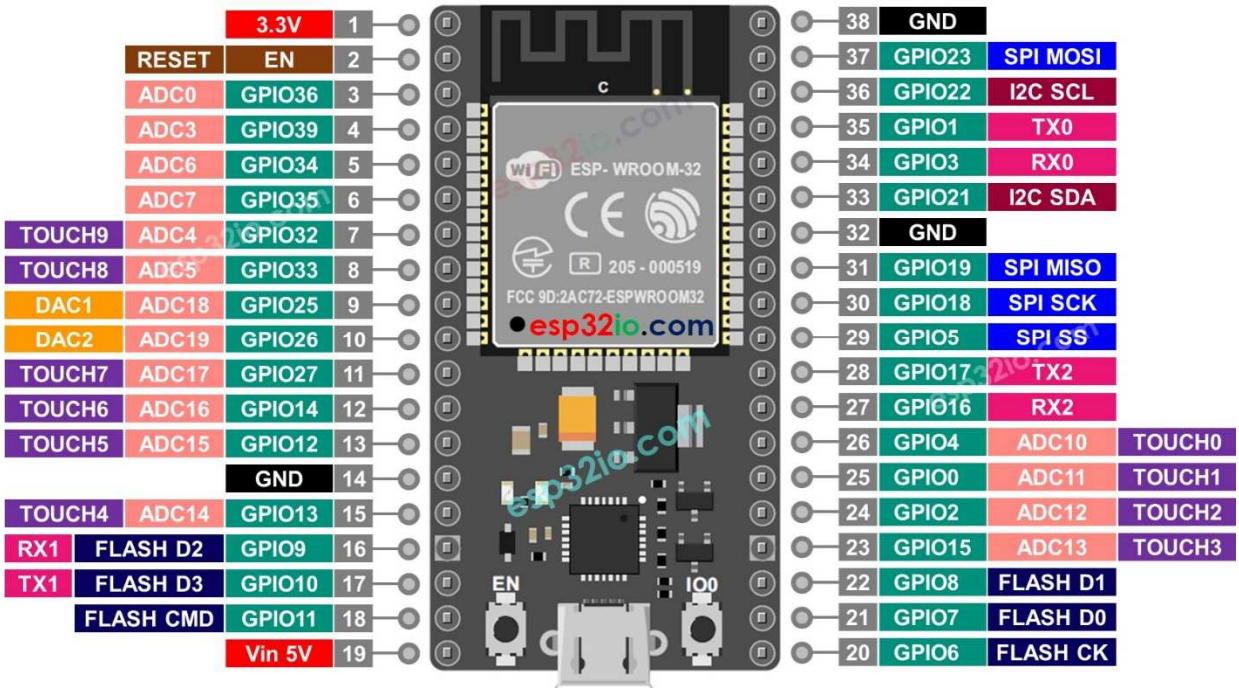

Figura 5 – Diagrama dos Pinos do ESP32

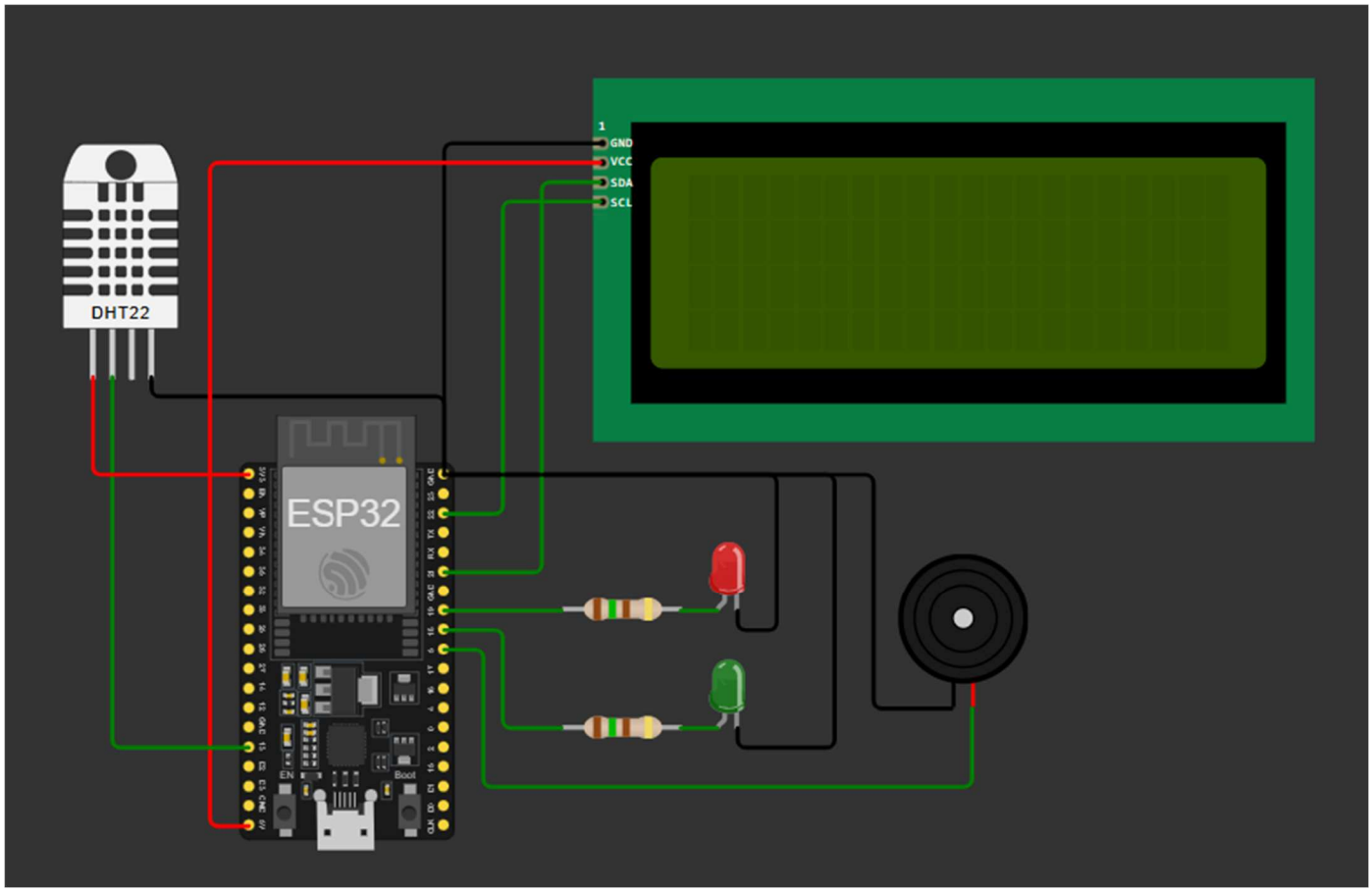

Figura 6 – Ligações dos Componentes

LCD 2004 (I2C):
GND→GND do ESP32
VCC→5V
SDA→GPIO 21 do ESP32
SCL→GPIO 22 do ESP32

LED Verde (LED_OK):
Anodo→Resistor 150 Ω→GPIO 18 do ESP32
Catodo→GND

LED Vermelho (LED_ALARME):
Anodo→Resistor 150 Ω→GPIO 19 do ESP32
Catodo→GND

Buzzer:
Terminal Positivo→GPIO 5 do ESP32
Terminal Negativo→GND

DHT22:

VCC→3.3V
SDA→GPIO 12 do ESP32
NC→não conectado
GND→GND

2.5. Estrutura do Código do Projeto

Após concluir a montagem e as ligações do circuito, insira o código do projeto (baseado em C/C++) no arquivo sketch.ino.

Em seguida, clique em Play (▶) para iniciar a simulação e visualizar o funcionamento do sistema.

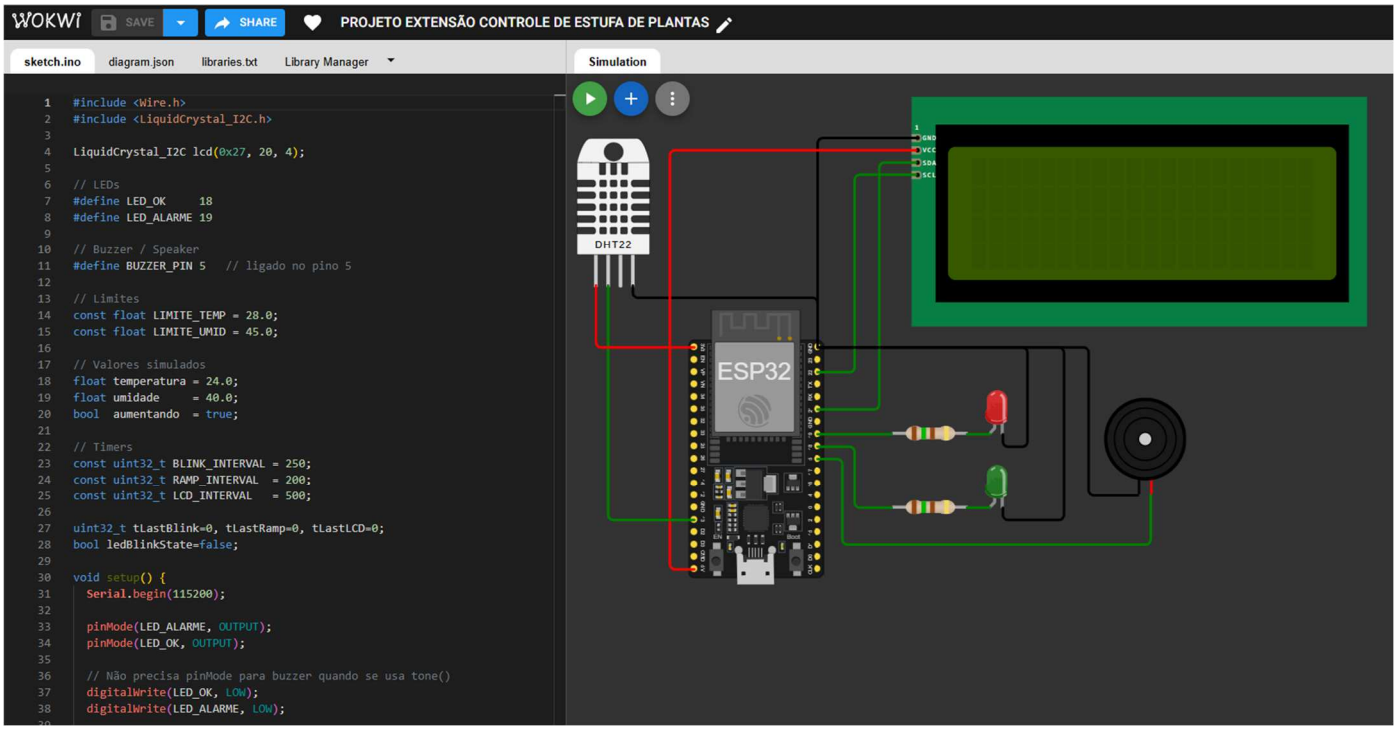

Figura 7 – Código do Projeto

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 20, 4);

// LEDs
#define LED_OK     18
#define LED_ALARME 19

// Buzzer / Speaker
#define BUZZER_PIN 5   // ligado no pino 5

// Limites
const float LIMITE_TEMP = 28.0;
```

```
const float LIMITE_UMID = 45.0;

// Valores simulados
float temperatura = 24.0;
float umidade     = 40.0;
bool  aumentando  = true;

// Timers
const uint32_t BLINK_INTERVAL = 250;
const uint32_t RAMP_INTERVAL  = 200;
const uint32_t LCD_INTERVAL   = 500;

uint32_t tLastBlink=0, tLastRamp=0, tLastLCD=0;
bool ledBlinkState=false;

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

  pinMode(LED_ALARME, OUTPUT);
  pinMode(LED_OK, OUTPUT);

  digitalWrite(LED_OK, LOW);
  digitalWrite(LED_ALARME, LOW);

  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(2,0); lcd.print("Simulacao  DHT22");
  lcd.setCursor(2,1); lcd.print("LCD + LEDS + SOM");
    lcd.setCursor(4,2); lcd.print("Iniciando...");
  delay(8000);
  lcd.clear();
}

void atualizaRampa() {
  if (aumentando) {
    temperatura += 0.1;
    umidade     -= 0.2;
    if (temperatura >= 30.0) aumentando = false;
  } else {
    temperatura -= 0.1;
    umidade     += 0.2;
    if (temperatura <= 19.0) aumentando = true;
  }
```

```
}

void mostraNoLCD(float t, float u, bool alarme, bool ok) {
  lcd.setCursor(0,0);
  lcd.print("LIMITE: TEMP E UMID ");

  lcd.setCursor(0,1);
  lcd.print("T:28.0");
  lcd.write((uint8_t)223);
  lcd.print("C  ");

  lcd.setCursor(9,1);
  lcd.print("U:45.0%");

  lcd.setCursor(0,2);
  lcd.print("T:");
  lcd.print(t,1);
  lcd.write((uint8_t)223);
  lcd.print("C  ");

  lcd.setCursor(9,2);
  lcd.print("U:");
  lcd.print(u,1);
  lcd.print("% ");

  lcd.setCursor(0,3);
  lcd.print("ALM:");
  lcd.print(alarme ? "ON " : "OFF");

  lcd.setCursor(9,3);
  lcd.print("SISTOK:");
  lcd.print(ok ? "YES " : "NO ");
}

void loop() {
  uint32_t now = millis();

  // Atualiza rampa
  if (now - tLastRamp >= RAMP_INTERVAL) {
    tLastRamp = now;
    atualizaRampa();
  }

  // Verifica alarme
```

```
  bool alarme = (temperatura > LIMITE_TEMP) ||
 (umidade > LIMITE_UMID);
  bool ok     = !alarme;

  // LED ALARME + SOM (estilo despertador)
  if (alarme) {
    if (now - tLastBlink >= BLINK_INTERVAL) {
      tLastBlink = now;
      ledBlinkState = !ledBlinkState;

      digitalWrite(LED_ALARME, ledBlinkState ? HIGH : LOW);

      if (ledBlinkState) {
        tone(BUZZER_PIN, 1000);   // 1000 Hz
      } else {
        noTone(BUZZER_PIN);
      }
    }
  } else {
    ledBlinkState = false;
    digitalWrite(LED_ALARME, LOW);
    noTone(BUZZER_PIN);
    tLastBlink = now;
  }

  // LED OK
  digitalWrite(LED_OK, ok ? HIGH : LOW);

  // LCD
  if (now - tLastLCD >= LCD_INTERVAL) {
    tLastLCD = now;
    mostraNoLCD(temperatura, umidade, alarme, ok);
  }
}
```

2.6. Explicação por Bloco de Código

2.6.1. Bibliotecas e Objeto do LCD

As bibliotecas Wire.h e LiquidCrystal_I2C.h permitem a comunicação com o Display LCD usando o protocolo I2C.

O objeto lcd(0x27, 20, 4) representa o Display de 20 colunas por 4 linhas no endereço I2C 0x27.

2.6.2. Definição de Pinos e Limites

São definidos os pinos dos LEDs e do Buzzer, além dos limites de

temperatura (28°C) e umidade (45%).

Esses limites representam a faixa considerada segura para a estufa de plantas na simulação.

2.6.3. Simulação de Temperatura e Umidade

As variáveis temperatura e umidade não são obtidas de um sensor físico.

Para fins didáticos, esses valores são simulados pela função atualizaRampa(), que ajusta gradualmente as medições ao longo do tempo: enquanto a temperatura aumenta, a umidade diminui até atingir um limite e, em seguida, o comportamento é invertido.

Essa variação controlada cria um ciclo de “aquecimento” e “resfriamento”, permitindo testar de forma consistente as condições de normalidade e alerta do sistema.

2.6.4. Timers e Uso de Millis()

Os timers BLINK_INTERVAL, RAMP_INTERVAL e LCD_INTERVAL definem de quanto em quanto tempo cada ação deve acontecer.

A função millis() fornece o tempo em milissegundos desde que o ESP32 foi ligado e, com isso, o programa verifica se já passou o intervalo necessário para atualizar a rampa, piscar o LED de alarme ou atualizar o LCD.

2.6.5. Função Setup()

No setup() são configurados os pinos dos LEDs, o estado inicial do Buzzer, a comunicação serial e o Display LCD.

Também é exibida uma tela de abertura com mensagens de início da simulação.

2.6.6. Função MostraNoLCD()

A função mostraNoLCD() organiza as informações nas quatro linhas do Display: limites de temperatura e umidade, valores atuais e o estado do sistema (alarme ligado/desligado e sistema OK ou não).

2.6.7. Função Loop() – Atualização e Decisão

Dentro de loop() o programa atualiza a simulação de temperatura e umidade, verifica se os valores ultrapassaram os limites, decide se está em alarme ou em condição normal e, conforme essa decisão, acende o LED Verde ou faz o LED Vermelho piscar junto com o Buzzer.

Por fim, atualiza o Display LCD em intervalos regulares.

2.7. Lógica de Funcionamento do Sistema

O sistema do projeto funciona da seguinte forma:

- O sistema mantém duas variáveis: temperatura e umidade, que são simuladas pela função atualizaRampa();
- A cada pequeno intervalo, a temperatura sobe ou desce um pouco e

a umidade faz o movimento inverso, imitando ciclos de aquecimento e resfriamento da estufa;

- São definidos limites de segurança: temperatura máxima de 28°C e umidade máxima de 45%.
- Se a temperatura passar de 28°C ou a umidade passar de 45%, o sistema entra em ALERTA: LED Vermelho pisca e Buzzer toca como um despertador, e no Display aparece ALM: ON e SISTOK: NO.
- Se os valores estiverem dentro dos limites, o sistema fica em OK: LED Verde aceso, LED Vermelho apagado, Buzzer desligado, e o Display mostra ALM: OFF e SISTOK: YES.
- O uso de timers com millis() permite atualizar a simulação, o LCD e o alarme em intervalos diferentes, dando a sensação de um sistema que trabalha em tempo real.

2.8. Link do Projeto

A seguir, disponibilizamos o link do projeto para acesso à simulação e visualização do protótipo em funcionamento.

https://wokwi.com/projects/448166584114793473

2.9. Conclusão do Projeto

O projeto Controle de Estufa de Plantas com IoT atingiu o objetivo de demonstrar, de forma simples e prática, como a Internet das Coisas pode ser aplicada ao monitoramento de condições ambientais. Embora as leituras de sensores tenham sido simuladas nesta etapa, a lógica implementada é compatível com aplicações reais, permitindo que o protótipo seja facilmente adaptado para uso em uma montagem física.

Além do aprendizado técnico em programação embarcada, eletrônica básica e uso de microcontroladores, o desenvolvimento do projeto também contribuiu para o aprimoramento de competências como trabalho em equipe, organização e produção de material educativo, em alinhamento com os objetivos da disciplina e da proposta da atividade.