Mini Projeto

Sistema de Irrigação

Alunos: Keven Garcia nº 30615
Francisco Oliveira nº 22252

Orientação: Prof. João Faria

ipvc estg

ERSC ENGENHARIA DE REDES E SISTEMAS DE COMPUTADORES

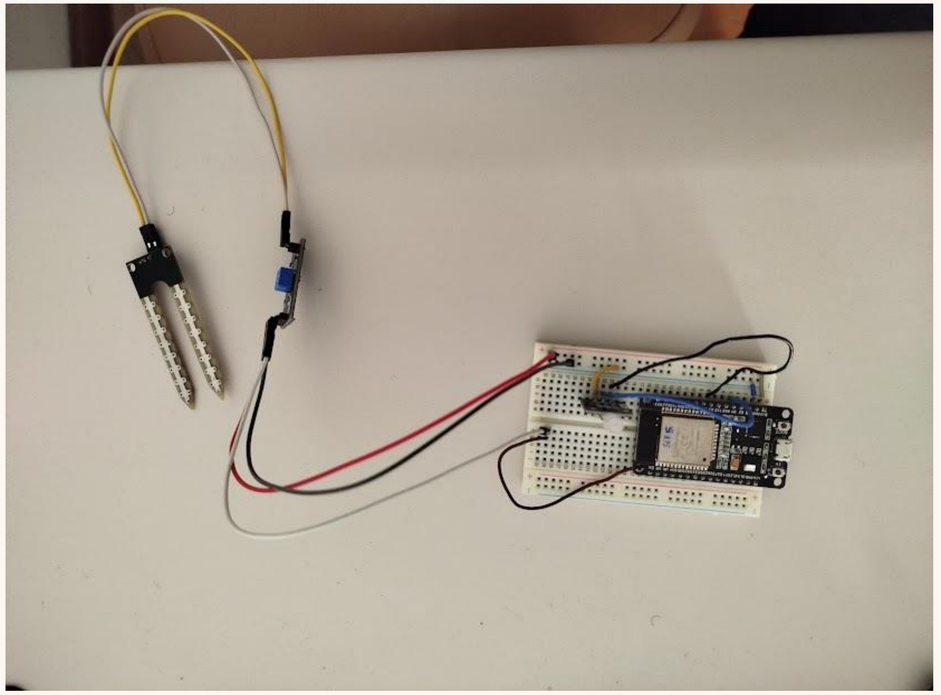
Introdução

- Nesta apresentação iremos demonstrar os métodos utilizados para atingir os nossos objetivos neste mini-projeto, tal como mostrar o setup do ESP32 e as ligações feitas para o bom funcionamento do nosso pequeno sistema IoT. Também será feita uma análise ao servidor HTTP e como funciona a troca de requests entre ESP32 e servidor HTTP. O nosso mini-projeto trata-se de um sistema de irrigação de controlo manual através de uma interface web. Através da web é possível analisar a humidade em tempo-real e ligar ou desligar o sistema de rega. Também são exibidos alertas consoante o nível de humidade do solo.

Setup Sensor e Atuador

O sensor está conectado ao módulo receptor que, por sua vez, está conectado ao ESP32. O GND do módulo deve ser conectado ao terminal GND da placa, o VCC à fonte de alimentação positiva, e a saída analógica (A0) ao pino 35 do microcontrolador.

O atuador (KY-011, LED de duas cores) tem o pino GND conectado ao terminal GND da placa, o pino do LED vermelho conectado ao pino D4, e o pino do LED verde conectado ao pino D5.



Configurações Iniciais

```
#include <WiFi.h>
#include <HTTPClient.h>

#define PIN_Sensor 35
#define LED_RED 4
#define LED_GREEN 5
```

```
const char* ssid = "MEO-F7C000";
const char* password = "786ea359e1";
const char* serverUrlSensor = "http://192.168.1.177:3000/sensor";    // Rota para enviar dados do sensor
const char* serverUrlControl = "http://192.168.1.177:3000/control";  // Rota para buscar estado de controle
```

```
void setup() {
  Serial.begin(115200);
  analogSetAttenuation(ADC_11db);
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);

  conectarWiFi();
}
```

```
void conectarWiFi() {
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Conectando ao Wi-Fi...");
  }
  Serial.println("Conectado ao Wi-Fi!");
}
```

Configuração ESP32

- O esp32 executa HTTP POST requests a cada 5 segundos com a percentagem de humidade no solo e GET requests a cada 2 segundos, requerendo o estado do sistema de rega (LED) enviado pelo servidor. O valor da humidade é primeiramente convertido em percentual de forma a ser facilmente interpretado pelo utilizador.

```
void verificarControle() {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverUrlControl);  // Conecta à rota de controle
    int httpResponseCode = http.GET();
    if (httpResponseCode > 0) {
      String response = http.getString();
      response.trim();  // Remove espaços em branco e quebras de linha
      Serial.print("Resposta do servidor: ");
      Serial.println(response);

      if (response == "on") {
        controlarLEDs("on");
      } else if (response == "off") {
        controlarLEDs("off");
      } else {
        Serial.println("Resposta desconhecida do servidor.");
      }
    } else {
      Serial.print("Erro ao verificar controle. Código de resposta: ");
      Serial.println(httpResponseCode);
    }
    http.end();
  } else {
    Serial.println("Wi-Fi desconectado. Não foi possível verificar controle.");
  }
}
```

HTTP GET request

```
void enviarDadosSensor() {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverUrlSensor);  // Conecta à rota de envio de sensor
    http.addHeader("Content-Type", "application/json");

    int humidade = ler_dados();
    String jsonData = "{\"humidity\": " + String(humidade) + "}";

    int httpResponseCode = http.POST(jsonData);
    if (httpResponseCode > 0) {
      Serial.print("Dados enviados com sucesso. Código de resposta: ");
      Serial.println(httpResponseCode);
    } else {
      Serial.println("Erro ao enviar dados do sensor.");
    }
    http.end();
  } else {
    Serial.println("Wi-Fi desconectado. Dados do sensor não enviados.");
  }
}
```

HTTP POST request

Configuração ESP32 (Continuação)

- A converção do valor de humidade é feita através da função map, onde é inserido o valor mínimo e máximo lido pelo sensor, respetivamente 4095 e 0. Este valor é posteriormente passado à função enviarDadosSensor(), responsável por gerar a POST request e enviá-la ao servidor, através da função ler_dados().

```
void enviarDadosSensor() {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverUrlSensor);  // Conecta à rota de envio de sensor
    http.addHeader("Content-Type", "application/json");

    int humidade = ler_dados();
    String jsonData = "{\"humidity\": " + String(humidade) + "}";
```

Uso dos dados convertidos na POST request

```
int ler_dados() {
  int valorBruto = analogRead(PIN_Sensor);
  int humidade = map(valorBruto, 4095, 0, 0, 100);
  return humidade;
}
```

Leitura e converção dos dados lidos pelo sensor

Servidor HTTP

O servidor foi configurado em Node.js utilizando o framework Express, com duas rotas principais: uma para receber dados e outra para enviar dados ao ESP32.

```
// Rota para receber dados do sensor
app.post('/sensor', (req, res) => {
  const { humidity } = req.body;

  if (humidity !== undefined) {
    const timestamp = new Date();
    humidityData.push({ time: timestamp, humidity });

    if (humidityData.length > 100) {
      humidityData.shift(); // Remove o dado mais antigo
    }

    console.log(`Humidade recebida: ${humidity}`);
    res.status(200).send('Dados do sensor recebidos com sucesso.');
  } else {
    res.status(400).send('Humidade não fornecida.');
  }
});
```

Rota que recebe os dados do sensor

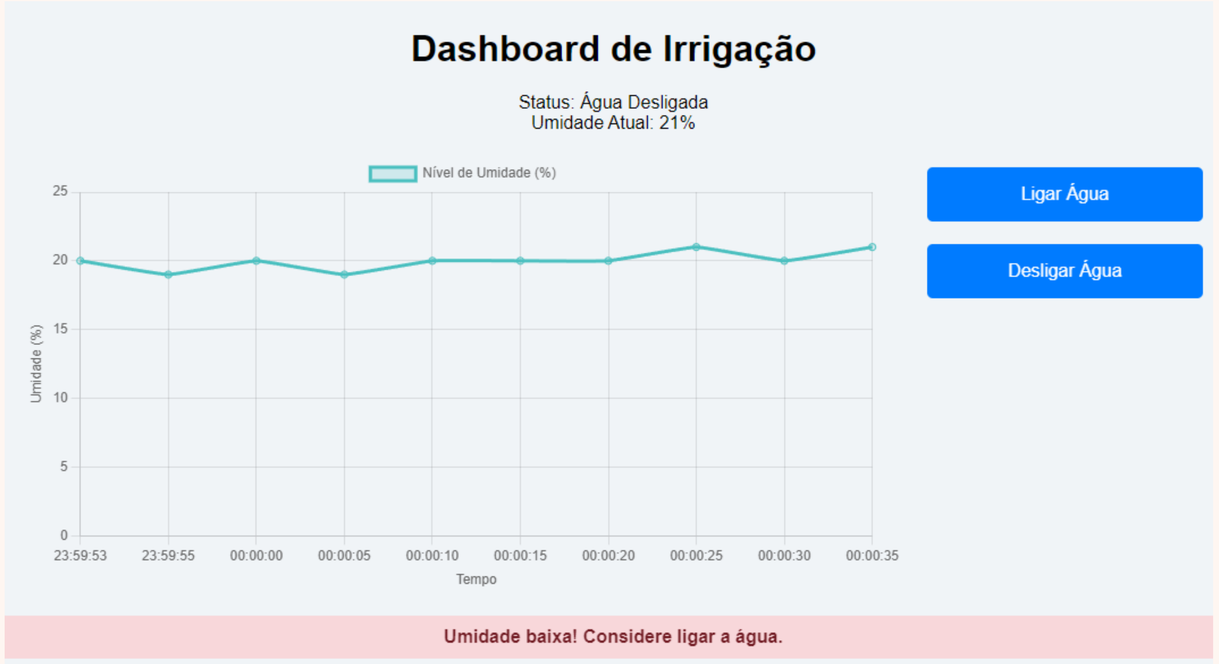
```
// Rota para controlar o sistema de irrigação
app.post('/control', (req, res) => {
  const { action } = req.body;

  if (action === 'on' || action === 'off') {
    waterStatus = action;
    console.log(`Estado da água atualizado para: ${action}`);
    res.status(200).send(`Ação de controle recebida: ${action}`);
  } else {
    res.status(400).send('Ação inválida.');
  }
});
```

Rota para controlar o led

Interface Web

- A interface web do servidor é responsável por exibir a percentagem de humidade num gráfico, mostrar alertas tendo em conta níveis pre-determindados de humidade e controlo manual do sistema de rega.



Demonstração em vídeo

Video de demonstração

Conclusão

O projeto desenvolvido demonstrou de forma prática como utilizar o ESP32 em conjunto com um sensor de humidade de solo para implementar um sistema básico de monitorização e controlo remoto. A leitura da humidade do solo foi integrada com sucesso a uma interface web dinâmica, que exibe os dados em formato gráfico, facilitando a análise dos dados.

Além disso, a funcionalidade de controle remoto do LED, que simula um sistema de rega, demonstrou a viabilidade de integrar funcionalidades de monitorização e automação em tempo real, reforçando o potencial do ESP32 em soluções IoT (Internet of Things). A implementação de alertas baseados na humidade medida destaca a sua aplicabilidade em cenários reais, como a gestão de sistemas de irrigação em pequenas culturas.

FIM

ipvc estg

ERSC ENGENHARIA DE REDES E SISTEMAS DE COMPUTADORES