FIAP - Faculdade de Informática e Administração Paulista

Nome do projeto:

Sistema de Irrigação Inteligente - FarmTech Solutions

Nome do grupo: Grupo 39

👩‍🎓 Integrantes:

- Fátima Vilela Candal – RM563003

👩‍🏫 Professores:

Tutor(a)

- Leonardo Ruiz Orabona

Coordenador(a)

- André Godoi Chiovato

📜 Descrição

Este projeto tem como objetivo desenvolver um **sistema inteligente de irrigação** utilizando sensores físicos simulados na plataforma **Wokwi**, um microcontrolador **ESP32 (Arduino)**, e um banco de dados **SQL** para armazenamento das leituras dos sensores.

O sistema pode **monitorar** a presença de nutrientes (Fósforo e Potássio), medir o **pH do solo**, analisar a **umidade do solo** e **controlar** a bomba de irrigação automaticamente com base nos dados coletados.

🗜️ Componentes do Sistema

- **ESP32** → Microcontrolador responsável pelo processamento dos dados com Framework Arduino.
- **Botões físicos** → Simulam sensores de Fósforo (P) e Potássio (K).
- **LDR (Light Dependent Resistor)** → Simula o sensor de pH do solo.
- **DHT22** → Sensor real para medir a umidade do solo.
- **Relé** → Controla a bomba de irrigação com base nos dados dos sensores.
- **Banco de dados SQL** → Armazena as leituras dos sensores para análises estatísticas.

💼 Estrutura de pastas

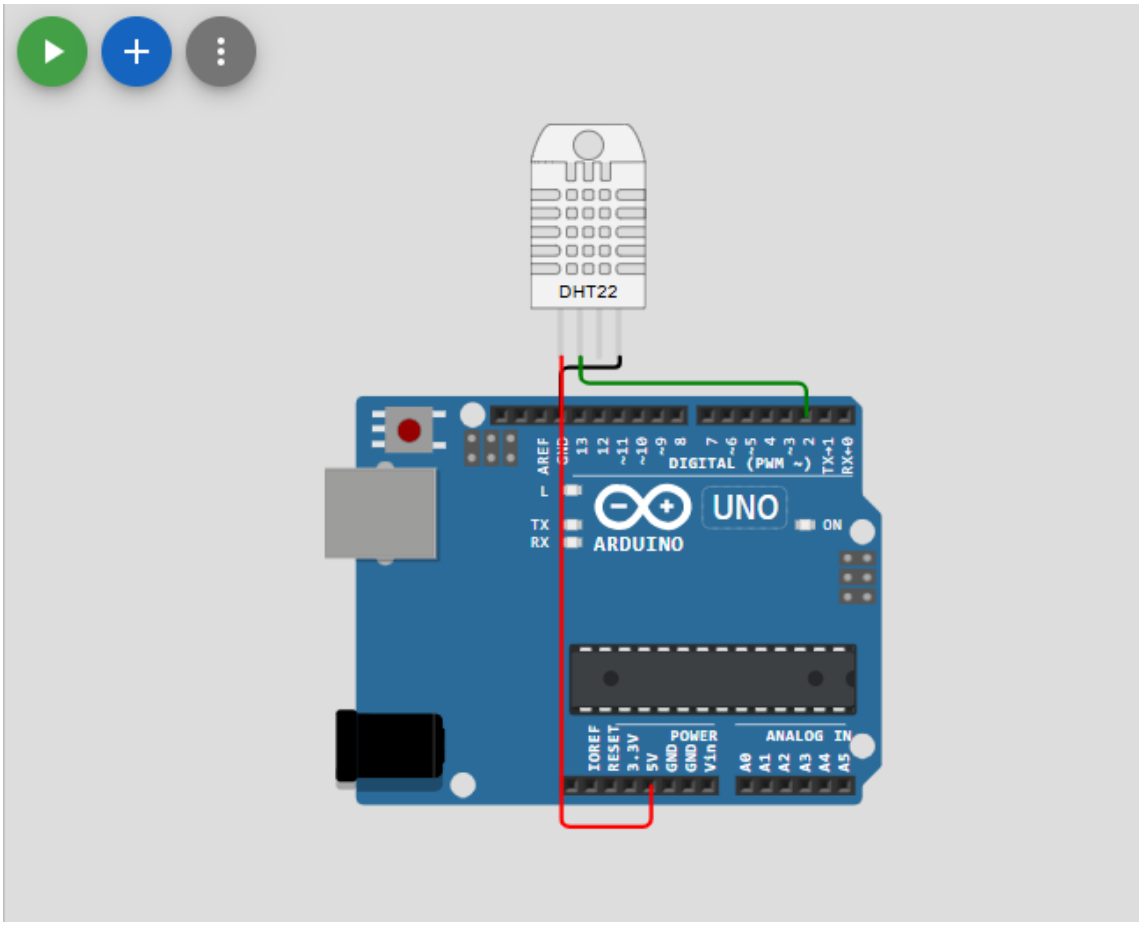
Os arquivos estão GITHUB no caminho:
https://github.com/rm563003/FIAP/tree/main/FASE%203%20-%20CAP%201/MeuProjeto

FIAP / FASE 3 – CAP 1 /

- **MeuProjeto**
    - **src:**
      Código fonte criado em C++: prog1.ino
      Código em Python: sensor.py
    - diagram.json
    - platformio.ini
    - wokwi.tmol
- **README.pdf**

🔧 Configuração do Circuito

A montagem dos sensores foi realizada no **VS Code**, utilizando as extensões **Wokwi e PlatformIO**. A imagem abaixo mostra a configuração do circuito montado na plataforma Wokwi:



🔧 Código - fonte

O código foi implementado em **C/C++** e pode ser acessado no seguinte repositório GitHub:

https://github.com/rm563003/FIAP/blob/main/FASE%203%20-%20CAP%201/MeuProjeto/src/prog1.ino

🔧 Lógica de funcionamento

1. Os sensores capturam os dados do solo.
2. O **ESP32 (Arduino)** processa os valores obtidos.
3. Se a umidade do solo for **menor que 40%**, a bomba de irrigação é **ativada automaticamente**.
4. Os dados coletados são armazenados no **banco de dados SQL** para consulta futura.

🔧 Estrutura do Banco de dados

Os dados dos sensores são armazenados no banco **sensor_data**, na tabela **sensor**, com a seguinte estrutura:

➿ Relacionamentos (DER)

Relacionamentos (DER)		
Relacionamento	Tipo	Observações
Sensor 🔁 Leitura	1:N	Um sensor pode gerar várias leituras
Cultura 🔁 Leitura	1:N	Uma cultura pode ter várias leituras
Cultura 🔁 Ajuste	1:N	Uma cultura pode ter vários ajustes
Ajuste 🔁 Sensor	N:M	Um ajuste pode usar vários sensores

--Entidade Sensor

```
CREATE TABLE sensor (

id INT AUTO_INCREMENT PRIMARY KEY,

timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,

phosphorus BOOLEAN,

potassium BOOLEAN,

ph_value FLOAT,

soil_humidity FLOAT

Tipo VARCHAR(50) NOT NULL,

Localizacao VARCHAR(100)

);
```

-- Entidade Cultura

```
CREATE TABLE Cultura (
   ID_Cultura INT PRIMARY KEY,
   Nome VARCHAR(50) NOT NULL,
   Data_Plantio DATE NOT NULL,
   Data_Colheita DATE
);
```

```
-- Entidade Leitura
CREATE TABLE Leitura (
    ID_Leitura INT PRIMARY KEY,
    Data_Hora DATETIME NOT NULL,
    Valor FLOAT NOT NULL,
    ID_Sensor INT,
    ID_Cultura INT,
    FOREIGN KEY (ID_Sensor) REFERENCES Sensor(ID_Sensor),
    FOREIGN KEY (ID_Cultura) REFERENCES Cultura(ID_Cultura)
);

-- Entidade Ajuste
CREATE TABLE Ajuste (
    ID_Ajuste INT PRIMARY KEY,
    Tipo_Ajuste VARCHAR(50) NOT NULL,
    Data_Hora DATETIME NOT NULL,
    Quantidade FLOAT NOT NULL,
    ID_Cultura INT,
    FOREIGN KEY (ID_Cultura) REFERENCES Cultura(ID_Cultura)
);


-- Entidade Ajuste Sensor
CREATE TABLE Sensor_Ajuste (
    ID_Sensor INT,
    ID_Ajuste INT,
    PRIMARY KEY (ID_Sensor, ID_Ajuste),
    FOREIGN KEY (ID_Sensor) REFERENCES Sensor(ID_Sensor),
    FOREIGN KEY (ID_Ajuste) REFERENCES Ajuste(ID_Ajuste)
);
```

🔧 Explicação da Estrutura

- id: Identificador único para cada leitura.
- timestamp: Registra automaticamente a data e hora de cada leitura.
- phosphorus **e** potassium: Armazena valores booleanos (presente = 1, ausente = 0).
- ph_value: Guarda o valor do pH do solo.
- soil_humidity: Armazena a porcentagem de umidade do solo.
- Tipo: Tipo do sensor.
- Localização: Localização do sensor.

🔧 Comandos CRUD

Para manipular os dados no banco SQL, o sistema pode executar as operações **Create, Read, Update e Delete**.

-- Inserção de Culturas Padrão

INSERT INTO Cultura (ID_Cultura, Nome, Data_Plantio, Data_Colheita) VALUES

(1, 'Soja', '2025-01-15', '2025-05-20'),

(2, 'Milho', '2025-02-10', '2025-06-15'),

(3, 'Cana de Açúcar', '2025-03-01', '2026-02-28'),

(4, 'Algodão', '2025-01-25', '2025-05-30'),

(5, 'Café', '2025-04-10', '2026-03-15'),

(6, 'Feijão', '2025-02-05', '2025-06-01'),

(7, 'Arroz', '2025-01-20', '2025-05-25');

-- Recuperação dos Dados:

Para visualizar os dados coletados, basta executar:

SELECT * FROM sensor ORDER BY timestamp DESC;

Isso retornará todas as leituras armazenadas, ordenadas da mais recente para a mais antiga.

Script Python (Simulação SQL)

Simulação de banco de dados SQL

https://github.com/rm563003/FIAP/blob/main/FASE%203%20-%20CAP%201/MeuProjeto/src/sensor.py

Licença