**FIAP - Faculdade de Informática e Administração Paulista**

[](https://www.fiap.com.br/)

**Nome do projeto**

**Nome do grupo**

**👨‍🎓 Integrantes:**

* [Nome do integrante 1](https://www.linkedin.com/company/inova-fusca)
* [Nome do integrante 2](https://www.linkedin.com/company/inova-fusca)
* [Nome do integrante 3](https://www.linkedin.com/company/inova-fusca)
* [Nome do integrante 4](https://www.linkedin.com/company/inova-fusca)
* [Nome do integrante 5](https://www.linkedin.com/company/inova-fusca)

**👩‍🏫 Professores:**

**Tutor(a)**

* [Nome do Tutor](https://www.linkedin.com/company/inova-fusca)

**Coordenador(a)**

* [Nome do Coordenador](https://www.linkedin.com/company/inova-fusca)

**📜 Descrição**

*Descreva seu projeto com base no texto do PBL (até 600 palavras)*

**📁 Estrutura de pastas**

Dentre os arquivos e pastas presentes na raiz do projeto, definem-se:

* **assets**: aqui estão os arquivos relacionados a elementos não-estruturados deste repositório, como imagens.
* **document**: aqui estão todos os documentos do projeto que as atividades poderão pedir. Na subpasta "other", adicione documentos complementares e menos importantes.
* **src**: Todo o código fonte criado para o desenvolvimento do projeto ao longo das 7 fases.
* **README.md**: arquivo que serve como guia e explicação geral sobre o projeto (o mesmo que você está lendo agora).

**🔧 Como executar o código**

*Acrescentar as informações necessárias sobre pré-requisitos (IDEs, serviços, bibliotecas etc.) e instalação básica do projeto, descrevendo eventuais versões utilizadas. Colocar um passo a passo de como o leitor pode baixar o seu código e executá-lo a partir de sua máquina ou seu repositório. Considere a explicação organizada em fase.*

**🗃 Histórico de lançamentos**

* 0.5.0 - XX/XX/2024 \*
* 0.4.0 - XX/XX/2024 \*
* 0.3.0 - XX/XX/2024 \*
* 0.2.0 - XX/XX/2024 \*
* 0.1.0 - XX/XX/2024 \*

**📋 Licença**

[MODELO GIT FIAP](https://github.com/agodoi/template) por [Fiap](https://fiap.com.br/) está licenciado sobre [Attribution 4.0 International](http://creativecommons.org/licenses/by/4.0/?ref=chooser-v1).

🚀 Sistema de Irrigação Inteligente - FarmTech Solutions

📌 Descrição

Este projeto tem como objetivo desenvolver um **sistema inteligente de irrigação** utilizando sensores físicos simulados na plataforma **Wokwi**, um microcontrolador **ESP32**, 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.
* **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.

🔧 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:

(Aqui você pode inserir a imagem do circuito .png gerada no Wokwi)

🖥️ Código-fonte

O código foi implementado em **C/C++** e pode ser acessado no seguinte repositório GitHub:  
🔗 [**Link para o código**](https://github.com/seu-repositorio)

🚦 Lógica de Funcionamento

1. Os sensores capturam os dados do solo.
2. O **ESP32** 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\_readings**, com a seguinte estrutura:

CREATE TABLE sensor\_readings ( id INT AUTO\_INCREMENT PRIMARY KEY, timestamp DATETIME DEFAULT CURRENT\_TIMESTAMP, phosphorus BOOLEAN, potassium BOOLEAN, ph\_value FLOAT, soil\_humidity FLOAT );

📝 Comandos CRUD

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

INSERT INTO sensor\_readings (phosphorus, potassium, ph\_value, soil\_humidity) VALUES (1, 0, 6.5, 38.2);



📖 Como Executar o Projeto

1️⃣ Clone o repositório do projeto:

git clone https://github.com/seu-repositorio.git

2️⃣ Instale as dependências no **PlatformIO**.  
3️⃣ Compile e envie o código para o **ESP32**.  
4️⃣ Monitore os dados dos sensores no **Monitor Serial**.  
5️⃣ Execute os scripts SQL para salvar os dados no banco.