

INSTITUTO FEDERAL

Paraíba

Projeto SE-2024.2

Equipe:

- Matheus Carneiro da Cunha
- Heitor Brunini
- Rafael Berg

Documentação

Estruturas Utilizadas:

- **IMUData**: Armazena os dados de aceleração e giroscópio obtidos do sensor.
- **Quaternion**: Representa os dados do quaternion calculados com base nos dados do sensor.
- **EulerAngle**: Contém os ângulos de Euler (roll, pitch e yaw) calculados a partir do quaternion.

imu_tools.h

Definições Iniciais

- **MPU6050_ADDR**: Define o endereço I2C padrão do sensor MPU6050.
- **SDA_PIN** e **SCL_PIN**: Definem os pinos GPIO usados para os sinais de dados (SDA) e clock (SCL) no barramento I2C.

Funções

imu_read_data Lê os dados do acelerômetro e giroscópio do sensor IMU, inicializando a comunicação I2C e coletando as informações necessárias.

- **Parâmetros**:
 - **IMUData *data**: Estrutura onde os dados lidos (aceleração e giroscópio) serão armazenados.

- Inicializa o sensor IMU com o endereço especificado (imu_init).
- Lê os dados de aceleração (imu_get_acceleration_data) e giroscópio (imu_get_gyroscope_data).
- Armazena os dados nas estruturas correspondentes (data->gyro e data->accel).
- Finaliza a comunicação com o sensor (imu_deinit).
- **Retorno:**
 - ESP_OK em caso de sucesso.
 - ESP_FAIL se algum passo falhar.

imu_calculate_quaternion Calcula o quaternion que representa a orientação do sensor a partir dos dados de aceleração e giroscópio.

- **Parâmetros:**
 - const IMUData *data: Estrutura contendo os dados do IMU.
 - Quaternion *quaternion: Estrutura onde o quaternion resultante será armazenado.
 - **Normalização do vetor de aceleração:**
Calcula a magnitude do vetor e divide cada componente (x, y, z) por essa magnitude. Isso garante que o vetor esteja no intervalo [0, 1].
 - **Conversão de dados do giroscópio:**
Converte os valores do giroscópio de radianos/s para graus/s.
 - **Cálculo dos ângulos de orientação:** Calcula o *pitch* (elevação), *roll* (inclinação) e *yaw* (giro).
 - **Cálculo do quaternion:**
Usa as fórmulas baseadas em seno e cosseno para combinar os ângulos e gerar o quaternion.
- **Retorno:**
 - ESP_OK em caso de sucesso.
 - ESP_FAIL se algum parâmetro for inválido ou ocorrer um erro nos cálculos.

imu_calculate_euler_angles Converte um quaternion em ângulos de Euler (roll, pitch, yaw), que são mais intuitivos para descrever a orientação de um objeto.

- **Parâmetros:**
 - const Quaternion *quaternion: Estrutura contendo o quaternion a ser convertido.
 - EulerAngle *euler: Estrutura onde os ângulos de Euler resultantes serão armazenados.
 - **Cálculo do roll (rotação ao longo do eixo X):** Usa as componentes w, x, y, e z do quaternion.
 - **Cálculo do pitch (rotação ao longo do eixo Y):** Calcula o arco seno do valor apropriado, garantindo que esteja no intervalo [-90°, 90°].

- **Cálculo do yaw (rotação ao longo do eixo Z):** Usa o arco tangente para calcular o ângulo de giro.
- **Retorno:**
 - ESP_OK em caso de sucesso.
 - ESP_FAIL se algum parâmetro for inválido.

sensor_imu.h

Funções

imu_init Inicializa o sensor MPU6050 configurando o driver I2C e verificando se o dispositivo está acessível.

Parâmetros:

- devAddr: Endereço I2C do dispositivo (normalmente 0x68 para o MPU6050).
- sda_pin: GPIO usado como linha de dados SDA.
- scl_pin: GPIO usado como linha de clock SCL.

Funcionamento:

- Configura o driver I2C no modo mestre com as definições de pinos e clock.
- Envia comandos I2C para verificar o registro WHO_AM_I no dispositivo para confirmar se o MPU6050 está conectado.
- Retorna ESP_OK se o sensor for encontrado e configurado com sucesso, ou ESP_ERR_NOT_FOUND em caso de falha.

imu_get_acceleration_data Lê os dados de aceleração do sensor.

Parâmetros:

- data: Estrutura onde os valores de aceleração (x, y, z) serão armazenados.

Funcionamento:

- Lê 6 bytes a partir do registro ACCEL_XOUT_H (dados brutos de aceleração para os eixos X, Y e Z).
- Converte os valores brutos para a unidade padrão de aceleração (m/s^2) e posteriormente para gravidade (g).
- Retorna ESP_OK em caso de sucesso ou ESP_FAIL se houver algum erro.

imu_get_gyroscope_data Lê os dados do giroscópio do sensor.

Parâmetros:

- data: Estrutura onde os valores do giroscópio (x, y, z) serão armazenados.

Funcionamento:

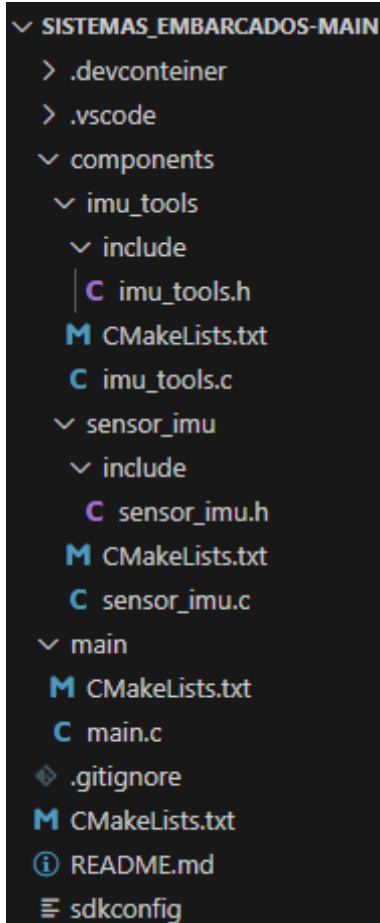
- Lê 6 bytes a partir do registro GYRO_XOUT_H (dados brutos do giroscópio para os eixos X, Y e Z).
- Converte os valores brutos para radianos por segundo (rad/s) e, posteriormente, para graus por segundo (°/s).
- Aplica divisões extras para ajustar a escala dos valores.
- Retorna ESP_OK em caso de sucesso ou ESP_FAIL em caso de erro.

imu_deinit Desinicializa o driver I2C, liberando os recursos alocados.

Funcionamento:

- Desinstala o driver I2C usando a função `i2c_driver_delete`.
- Retorna ESP_OK se a operação for bem-sucedida ou ESP_FAIL em caso de falha.

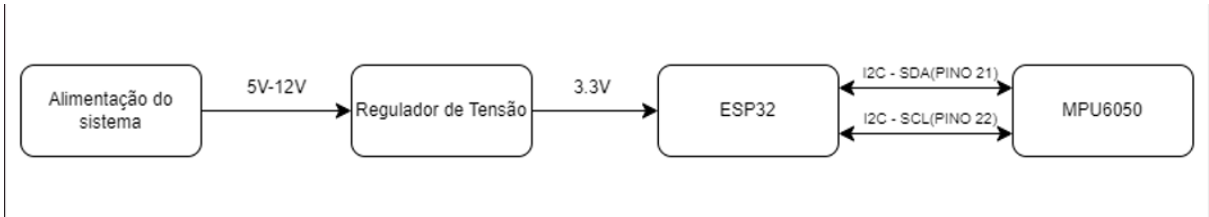
Estrutura de pastas:



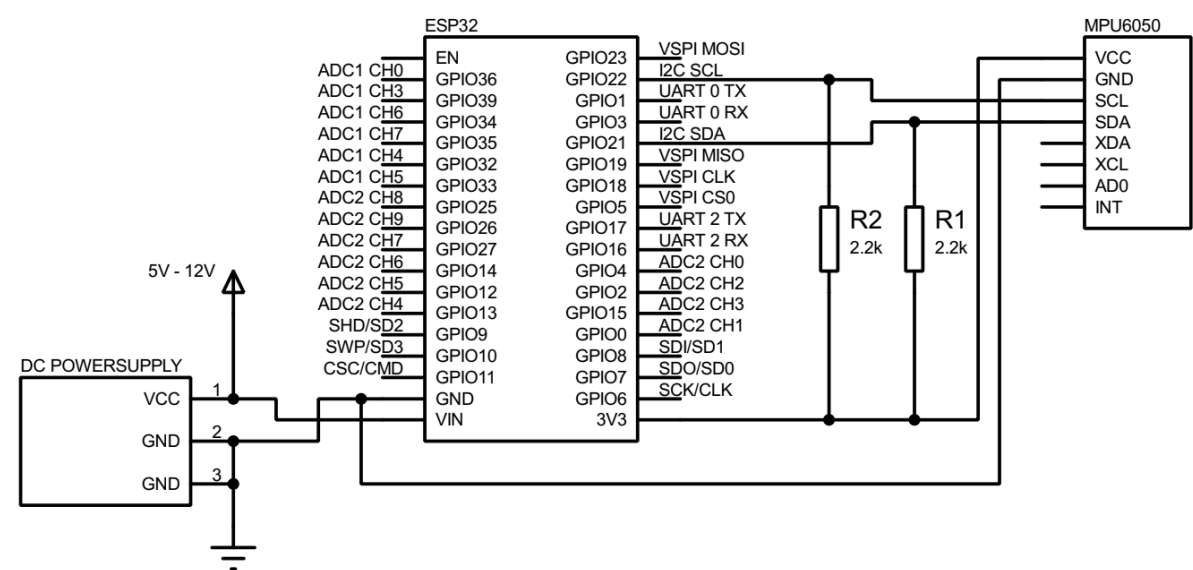
```

└─ SISTEMAS_EMBARCADOS-MAIN
   ├── .devcontainer
   ├── .vscode
   ├── components
   │   ├── imu_tools
   │   │   ├── include
   │   │   │   └─ imu_tools.h
   │   │   └─ CMakeLists.txt
   │   └─ imu_tools.c
   ├── sensor_imu
   │   ├── include
   │   │   └─ sensor_imu.h
   │   └─ CMakeLists.txt
   ├── sensor_imu.c
   ├── main
   │   ├── CMakeLists.txt
   │   └─ main.c
   ├── .gitignore
   ├── CMakeLists.txt
   ├── README.md
   └─ sdkconfig
```

Diagrama de Blocos do Protótipo do Hardware



Esquemático Elétrico do Hardware



Arquitetura

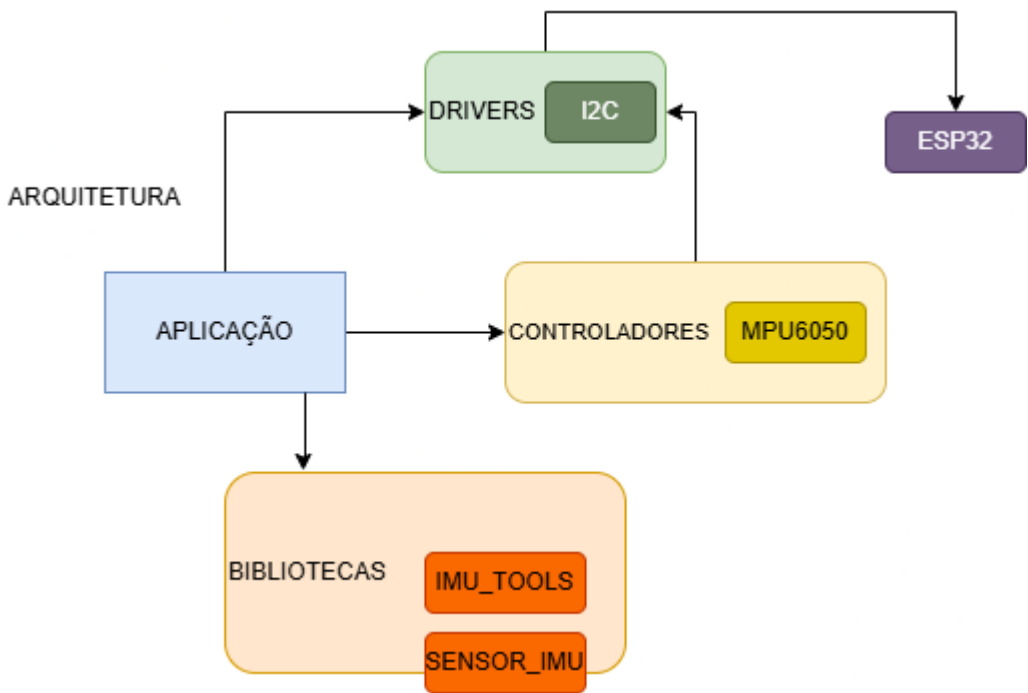


Diagrama de estados

