



Relatório final da disciplina de Sistemas Embarcados 2024.1

Luiz Eduardo Bronzeado Pessoa
Luiz Henrique da Silva Oliveira
Victor Hugo de Oliveira Gangorra

Setembro de 2024

Etapa 1

<https://github.com/luizh-silva-oliveira/projeto-sistemas-embarcados>

Nesta etapa foram desenvolvidas duas bibliotecas específicas para o uso do módulo **MPU6050** um sensor que combina um acelerômetro de três eixos e um giroscópio de três eixos em um único componente. Essas bibliotecas foram projetadas para facilitar a aquisição e o processamento de dados provenientes do acelerômetro e do giroscópio, permitindo uma leitura precisa e eficiente das medições de aceleração e rotação em diferentes eixos.

Foram criadas duas bibliotecas, a `sensor_imu` e a `imu_tools`. A `sensor_tools` basicamente serviu para as definições das structs que capturam os dados do acelerômetro e do giroscópio. Já a `imu_tools` serviu para guardar os dados do sensor, dos quaternions e dos ângulos de euler.

Funções da **sensor_imu**:

imu_init(): Configura a comunicação I2C e o MPU6050 deixando ele pronto para captura dos dados.

get_acceleration_data(AccelerationData *data): Retorna os dados do acelerômetro nos eixos x, y e z.

get_gyroscope_data(GyroscopeData *data): Retorna os dados do giroscópio nos eixos x, y e z.

Funções da **imu_tools**:

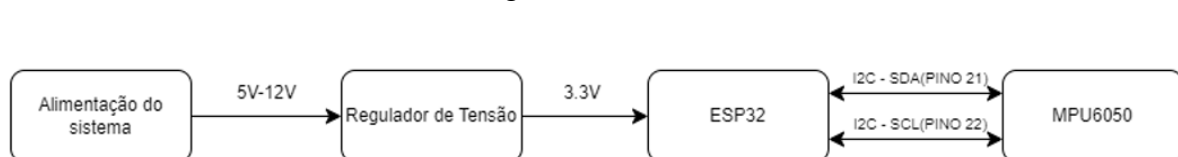
get_imu_data(IMUData *data): Captura os dados do acelerômetro e do giroscópio e guarda no struc IMUData passado como parâmetro.

calculate_quaternion(const IMUData *data, Quaternion *quaternion): A partir dos dados guardados na estrutura IMUData vai calcular os quatro quaternions.

quaternion_to_euler(const Quaternion *quaternion, EulerAngle *euler): A partir dos dados guardados na estrutura Quaternion vai calcular os três ângulos de euler yaw, pitch e roll.

get_quaternion(Quaternion *quaternion): Vai pegar o Quaternion calculado na função `calculate_quaternion` e gravar no quaternion passado como parâmetro.

Diagrama de blocos:



Esquemático:

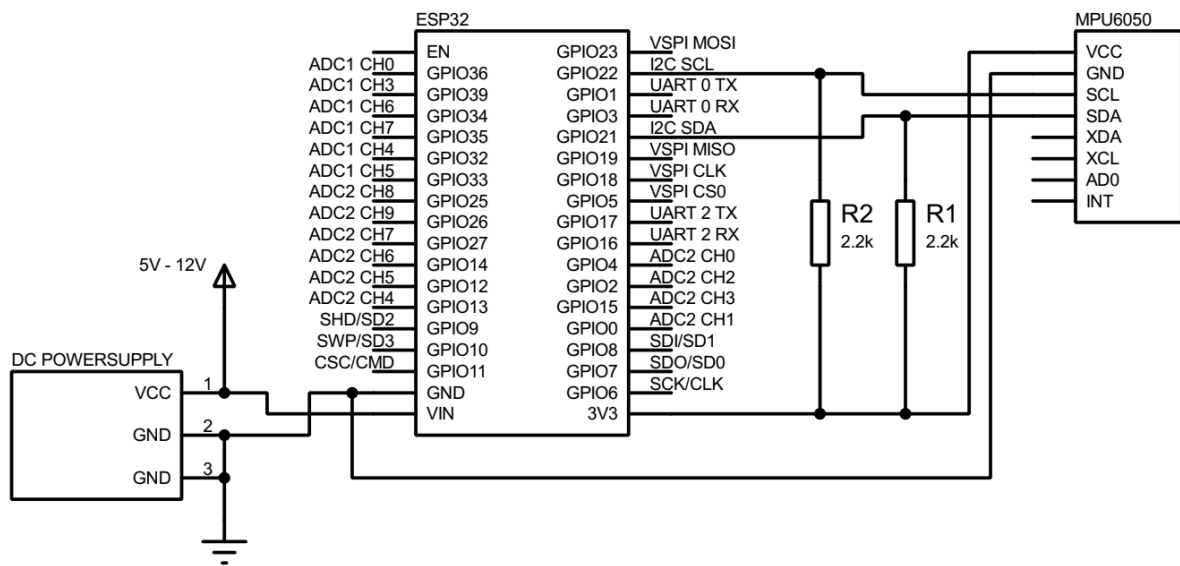
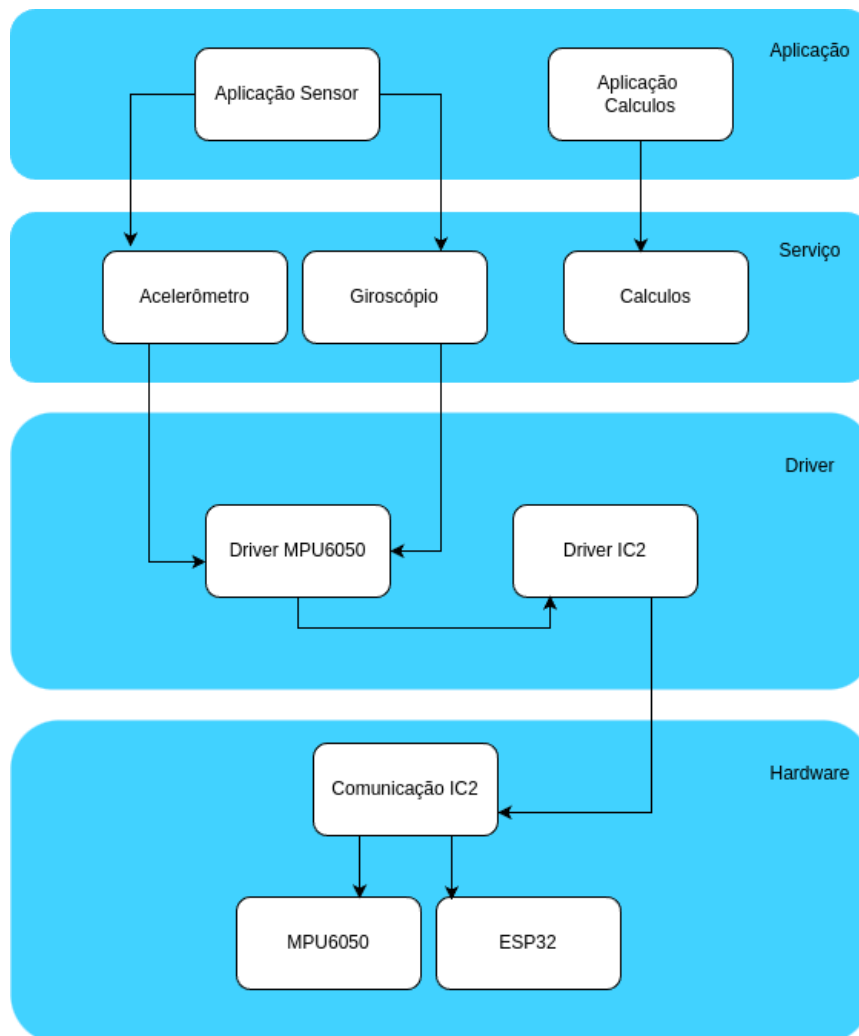


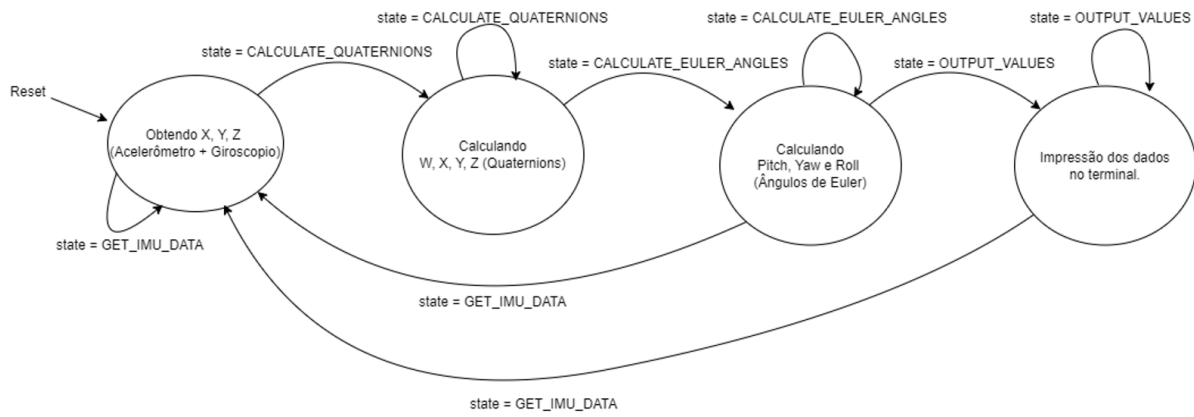
Diagrama de Arquitetura:



Vídeo Explicativo:

Link: <https://www.youtube.com/watch?v=z6EThjjiCBM>

Máquina de estados:



Etapa 2

<https://github.com/luizh-silva-oliveira/projeto-sistemas-embarcados-servo?autouser=0>

Nesta etapa foram desenvolvidas duas bibliotecas específicas para o controle de **servo motores**, facilitando a interface entre os sistemas de controle e os atuadores mecânicos. Essas bibliotecas permitem a manipulação precisa do posicionamento angular dos servos, e deixá-los disponíveis para serem utilizados.

Das duas bibliotecas, a servo hw basicamente serviu como interface de baixo nível para configurar e controlar servos usando ESP32 PWM. Já a servo_tools serviu como biblioteca auxiliar da servo_hw, com funções de inicialização e de controle dos ângulos dos servo motores.

Configurações da **servo hw**:

ServoConfig: Define parâmetros como ângulo máximo, largura de pulso mínima e máxima, frequência PWM, canal LEDC, ciclo de trabalho e pino do servo.

ServoAngle: Armazena o ângulo atual do servo.

Funções da **servo hw**:

hw_servo_init(uint8_t gpio_num): Inicializa o servo configurando o PWM no pino especificado.

hw_servo_set_pulse_width(uint8_t gpio_num, uint32_t pulse_width_us): Define a largura do pulso PWM para controlar a posição do servo.

hw_servo_deinit(uint8_t gpio_num): Desativa o servo e desabilita o PWM no pino especificado.

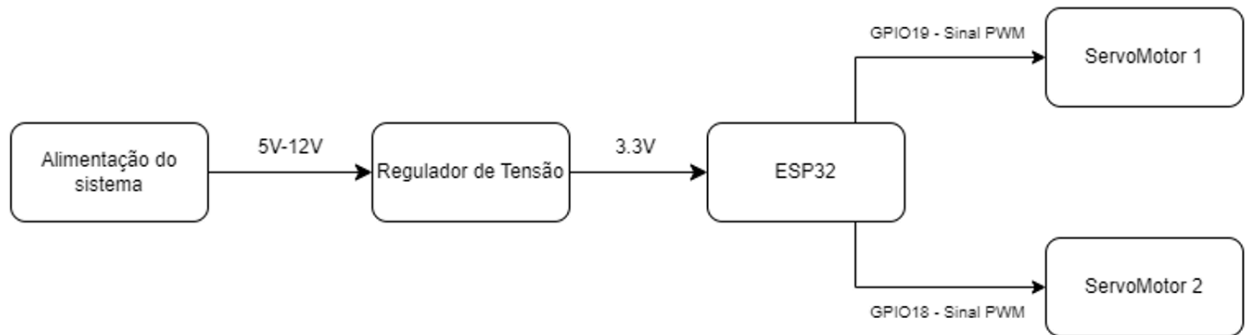
Funções da **servo_tools**:

servo_init(ServoConfig *config): Inicializa o servo usando as configurações especificadas na estrutura ServoConfig.

servo_set_angle(ServoConfig *config, ServoAngle angle): Define o ângulo do servo com base na estrutura ServoAngle.

servo_get_angle(const ServoConfig *config, ServoAngle *angle): Obtém o ângulo atual do servo e o armazena na estrutura ServoAngle.

Diagrama de blocos:



Esquemático:

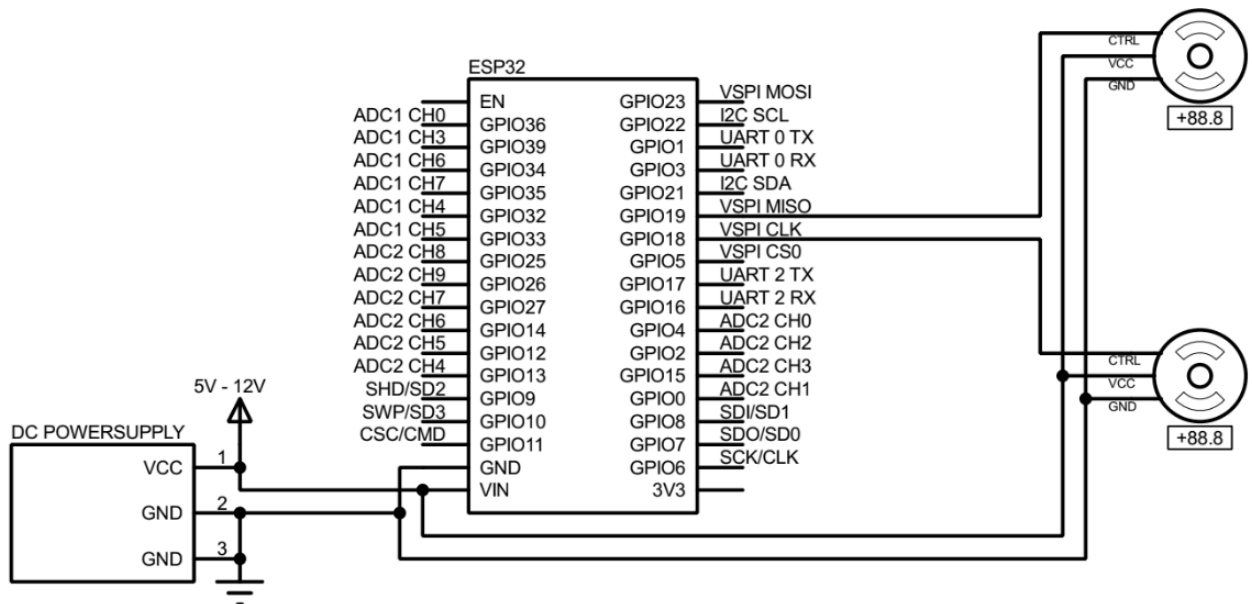
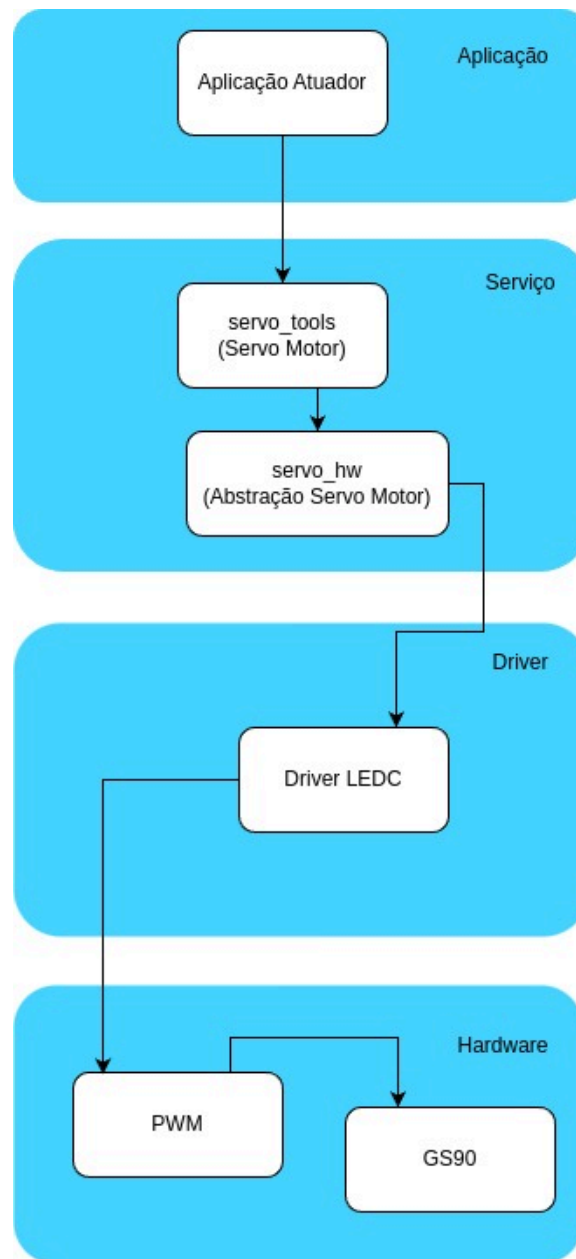


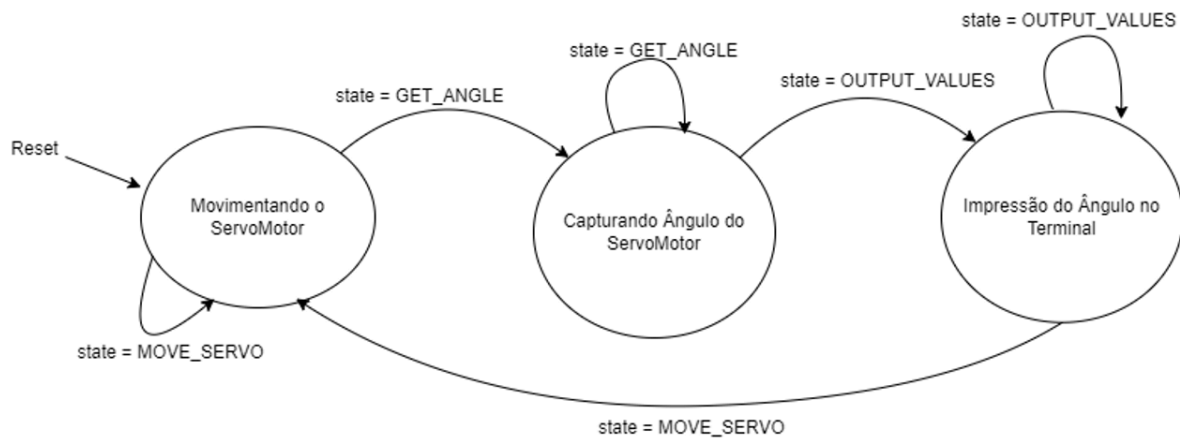
Diagrama de Arquitetura:



Vídeo Explicativo:

Link: <https://www.youtube.com/watch?v=dHcb04kcQVU>

Máquina de estados:

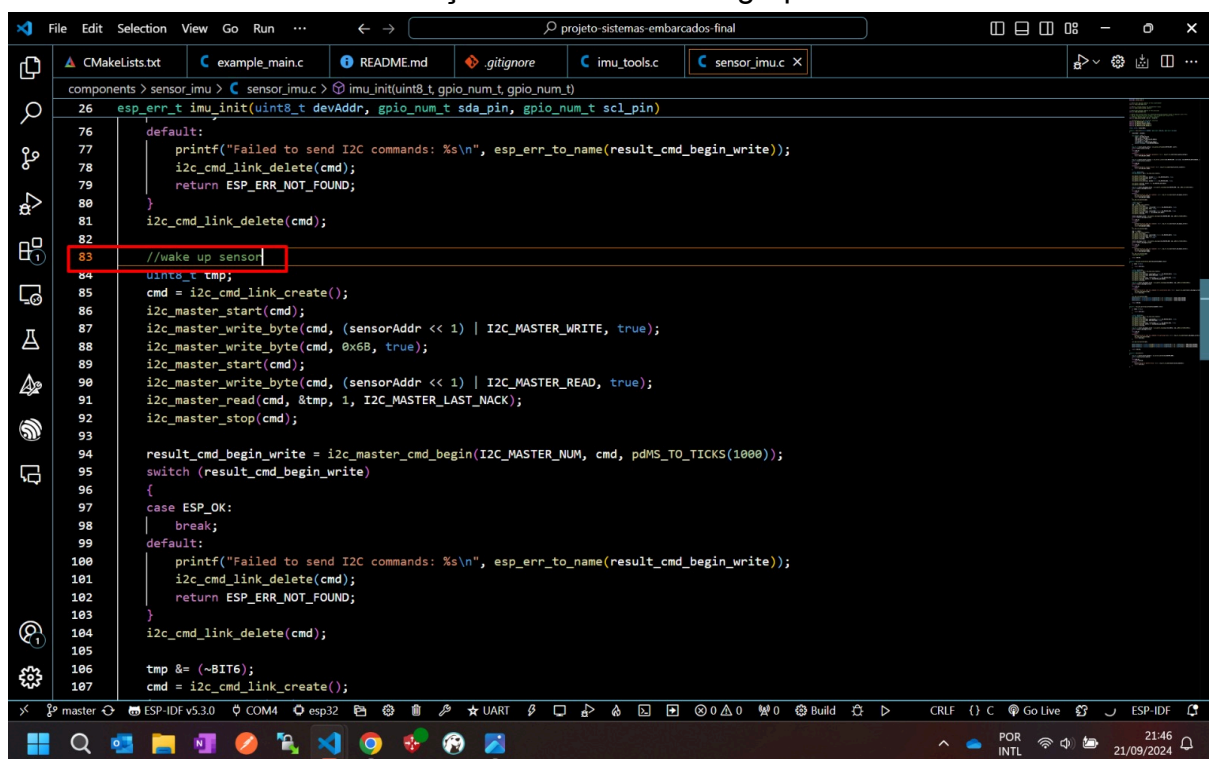


Etapa 3

<https://github.com/luizh-silva-oliveira/Servo-Axis-Control>

Nesta etapa foi desenvolvido um projeto que utiliza o sensor **MPU6050** para controlar dois **servo motores**, onde um servo representa o ângulo **roll** (inclinação lateral) e o outro o ângulo **pitch** (inclinação frontal/traseira). Utilizando as funções e configurações desenvolvidas nas duas outras etapas, foi possível utilizar os dados do giroscópio e acelerômetro capturados e armazenados na etapa 1 e atrelar esses dados às configurações e funções de manipulação dos servos da etapa 2.

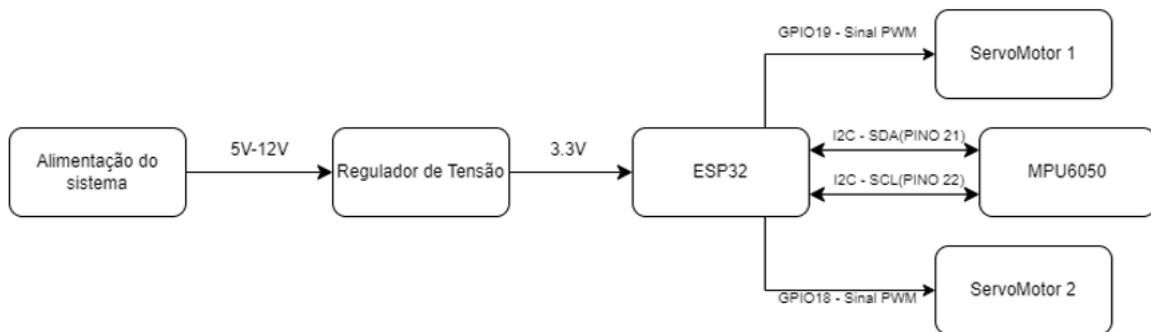
Avaliação da biblioteca do grupo 3:



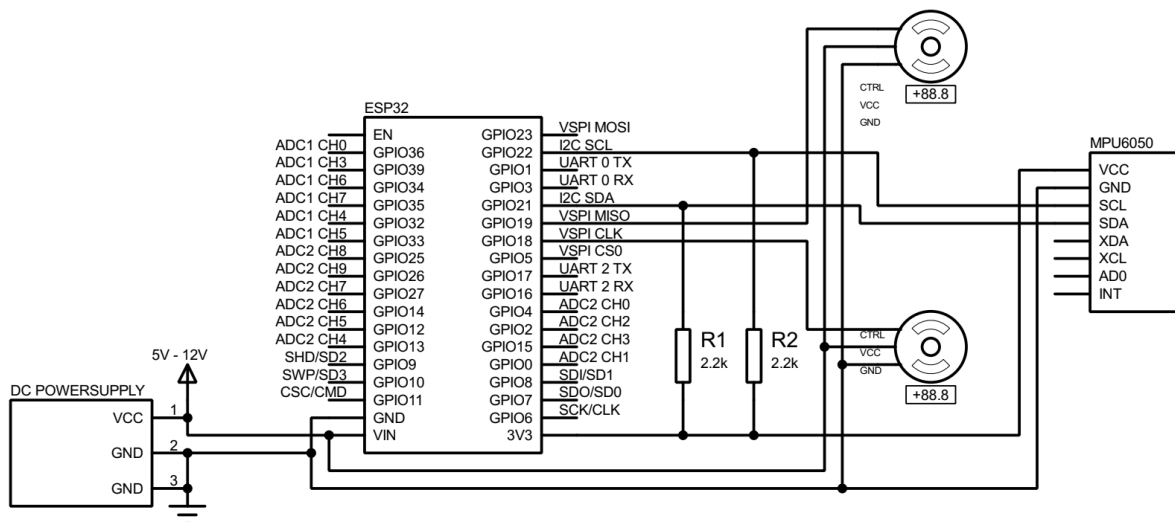
```
26 esp_err_t imu_init(uint8_t devAddr, gpio_num_t sda_pin, gpio_num_t scl_pin)
{
    default:
        printf("Failed to send I2C commands: %s\n", esp_err_to_name(result_cmd_begin_write));
        i2c_cmd_link_delete(cmd);
        return ESP_ERR_NOT_FOUND;
    }
    i2c_cmd_link_delete(cmd);
}
82
83 //wake up sensor
84 uint8_t tmp;
85 cmd = i2c_cmd_link_create();
86 i2c_master_start(cmd);
87 i2c_master_write_byte(cmd, (sensorAddr << 1) | I2C_MASTER_WRITE, true);
88 i2c_master_write_byte(cmd, 0x6B, true);
89 i2c_master_start(cmd);
90 i2c_master_write_byte(cmd, (sensorAddr << 1) | I2C_MASTER_READ, true);
91 i2c_master_read(cmd, &tmp, 1, I2C_MASTER_LAST_NACK);
92 i2c_master_stop(cmd);
93
94 result_cmd_begin_write = i2c_master_cmd_begin(I2C_MASTER_NUM, cmd, pdMS_TO_TICKS(1000));
95 switch (result_cmd_begin_write)
96 {
97     case ESP_OK:
98         break;
99     default:
100         printf("Failed to send I2C commands: %s\n", esp_err_to_name(result_cmd_begin_write));
101         i2c_cmd_link_delete(cmd);
102         return ESP_ERR_NOT_FOUND;
103     }
104     i2c_cmd_link_delete(cmd);
105
106     tmp &= (~BIT6);
107     cmd = i2c_cmd_link_create();
```

Foi necessário fazer uma mudança pontual na biblioteca **sensor_imu** do outro grupo, onde havia um problema da linha 83 até a linha 125. Basicamente o estado do sensor não estava sendo alterado do modo de hibernação, e nesse modo o sensor entra em um estado de economia de energia, assim não ocorre a captura dos dados.

Diagrama de blocos:



Esquemático:



Vídeo Demonstrativo:

Link: <https://www.youtube.com/shorts/DxNH4GY1BXM?feature=share>

Máquina de estados:

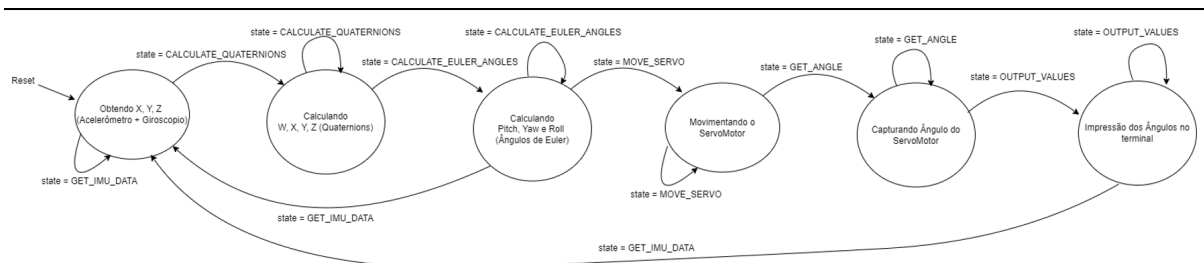


Diagrama de arquitetura:

