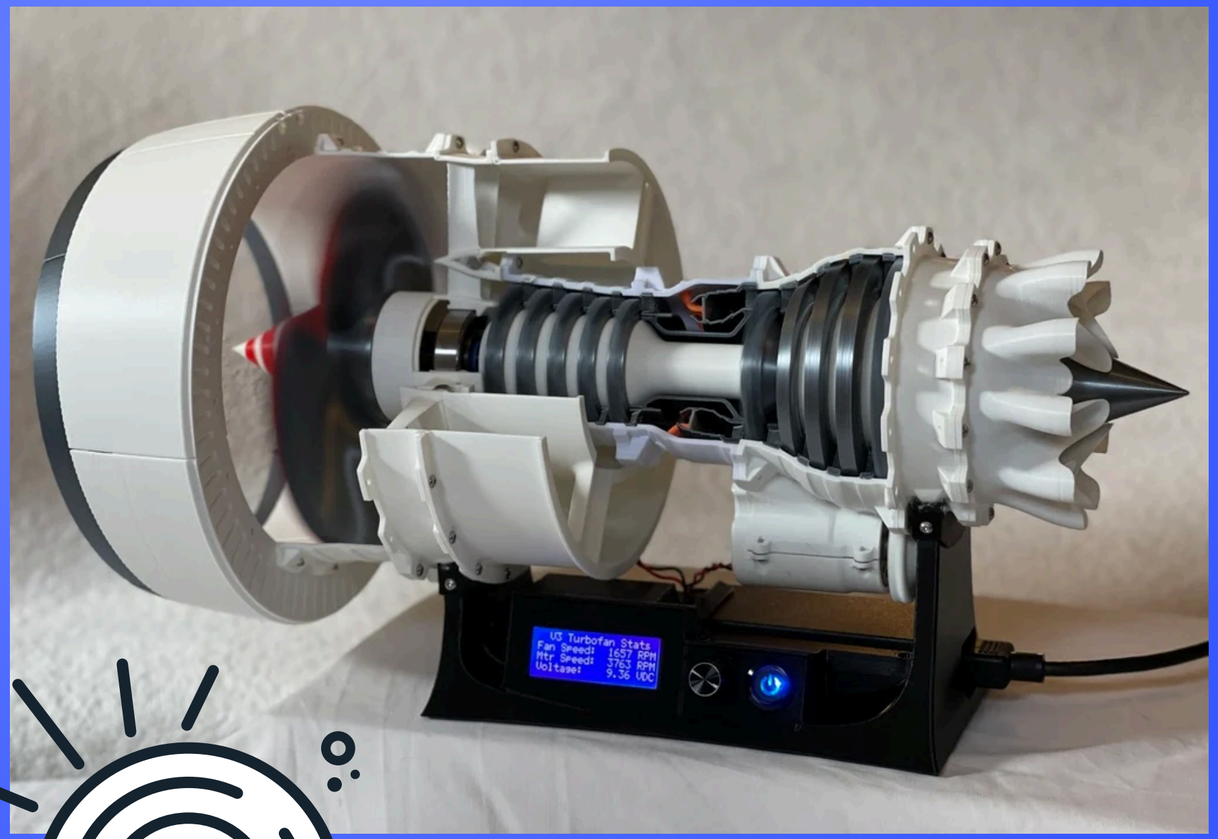
Projeto:
Motor Elétrico com Controle Web

1. Definição do Problema



Introdução

O projeto visa desenvolver um motor cuja rotação possa ser controlada e monitorada em tempo real por meio de um aplicativo web.

Importância

- Controle de RPM fundamental para performance de motores
- Permite regular consumo de energia / combustível
- Impacto ambiental positivo: menos emissões com rotações adequadas
- Aprimoramento de habilidades práticas em sistemas embarcados

Motivação

- Desenvolver uma aplicação com sistemas embarcados
- Aprender controle de motores e aplicações IoT
- Explorar a integração entre hardware e software

OBJETIVOS PRINCIPAIS

- Controlar velocidade de rotação do motor via aplicativo web
- Medir e visualizar métricas de rotação em tempo real
- Permitir acesso público controlado via WiFi
- Demonstrar integração hardware-software

OBJETIVOS ESPECÍFICOS

- Controlar velocidade do motor via PWM no ESP32
- Medir RPM com sensor de rotação (Hall Effect ou óptico)
- Desenvolver interface web para controle e métricas
- Criar comunicação bidirecional (WebSocket ou MQTT) entre App e ESP32

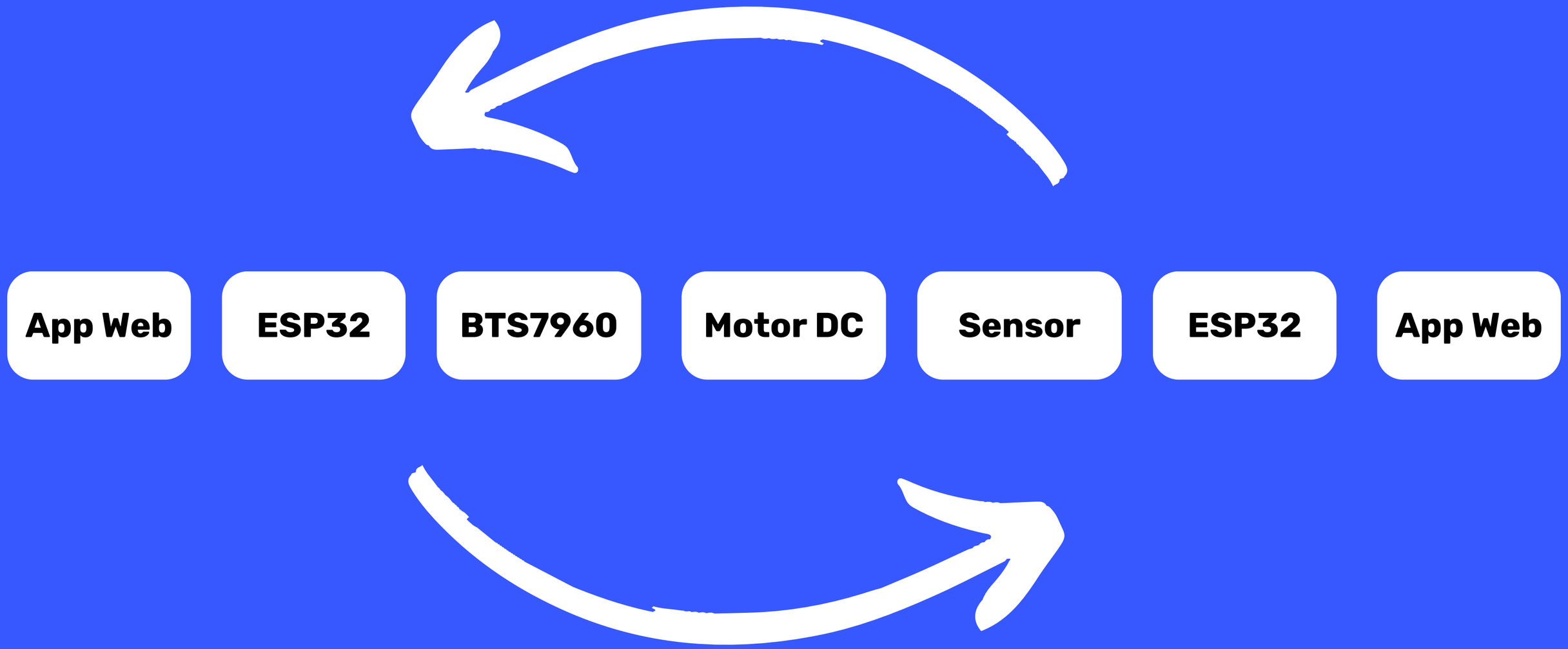
Funcionalidades

O projeto vai permitir:

- Controle de velocidade do motor em tempo real via aplicativo web.
- Visualização de métricas do motor, como RPM, potência estimada, consumo de energia e tempo de operação.
- Acesso público controlado via Wi-Fi, incluindo sistema de fila e QR Code para acesso rápido.
- Controle bidirecional do motor (rotação horária e anti-horária) com proteção integrada do driver.
- Registro e histórico das métricas para análise posterior de desempenho e eficiência energética.

Arquitetura Utilizada

- Microcontrolador (MCU): ESPC6-32 N4.
- Driver do motor: BTS7960 H-Bridge (aplica PWM e protege o motor).
- Motor: DC GA25-370 (169 RPM, 6V).
- Sensor de rotação: Hall Effect (KY-003) ou sensor óptico para medir RPM.
- Barramento de comunicação:
- Entre App Web e ESP32: Wi-Fi usando WebSocket (ou MQTT) para comunicação bidirecional em tempo real.
- Entre ESP32 e driver/motor: sinais PWM e controle digital de direção.


App Web
ESP32
BTS7960
Motor DC
Sensor
ESP32
App Web

Metodologia e Organização do Grupo

Metodologia de desenvolvimento:

- Hardware: cascata (Waterfall) → montagem, testes e integração do motor, driver e sensor.
- Software: SCRUM → desenvolvimento do frontend, backend, comunicação e interface de métricas.

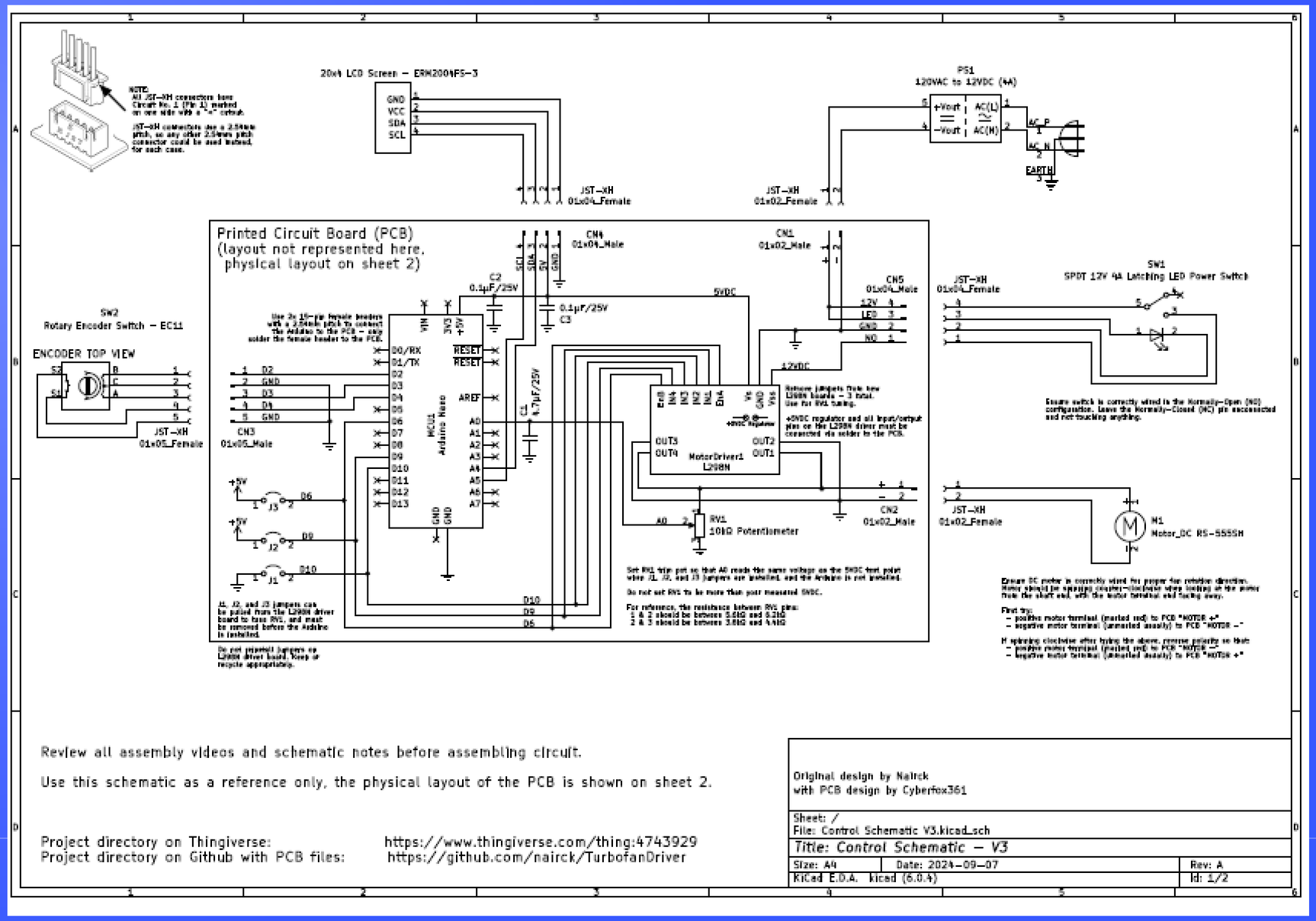
Organização do time:

- Equipe Hardware: controla motor, sensores, driver e alimentação, faz ajustes físicos e testes de integração.
- Equipe Software: desenvolve aplicativo web, interface de controle, gráficos de métricas, sistema de usuários e comunicação com ESP32.

Integração Hardware-Software: fases de teste conjunto, validação de controle do motor e visualização de métricas em tempo real.

→

Parte Prática - ETAPA 1:



Parte Prática - ETAPA 1:

OBRIGATÓRIO:

- 1x Capacitor eletrolítico 470µF/25V
- 2x Capacitor cerâmico 100nF
- 1x Fusível 1A-2A lâmina
- 1x Porta-fusível para lâmina
- 20x Jumpers macho-macho
- 10x Jumpers macho-fêmea
- 1x Protoboard 830 pontos
- 2x Step-downs
- Depois Hall Effect (KY-003)

-Alimentação: Bateria 12 volts

Consumo total: ~1A

Autonomia: ~2 horas contínuas

Perfeitamente adequado para demonstrações!

OPCIONAL (facilita):

- 2x JST-XH 2 pinos (bateria/motor)
- 1x JST-XH 4 pinos (BTS7960)
- Fio 22AWG colorido (2m)

Parte Prática - ETAPA 1:

sketch.ino | diagram.json | Library Manager

```ino
int motor1Pin1 = 27;
int motor1Pin2 = 26;
int enable1Pin = 14;

void setup() {
  pinMode(motor1Pin1, OUTPUT);
  pinMode(motor1Pin2, OUTPUT);
  pinMode(enable1Pin, OUTPUT);

  Serial.begin(115200);
}

void loop() {
  // Motor para frente
  Serial.println("Motor frente");
  digitalWrite(motor1Pin1, LOW);
  digitalWrite(motor1Pin2, HIGH);
  digitalWrite(enable1Pin, HIGH); // velocidade máxima
  delay(2000);

  // Motor parado
  Serial.println("Motor parado");
  digitalWrite(motor1Pin1, LOW);
  digitalWrite(motor1Pin2, LOW);
  digitalWrite(enable1Pin, LOW);
  delay(1000);

```

Simulation

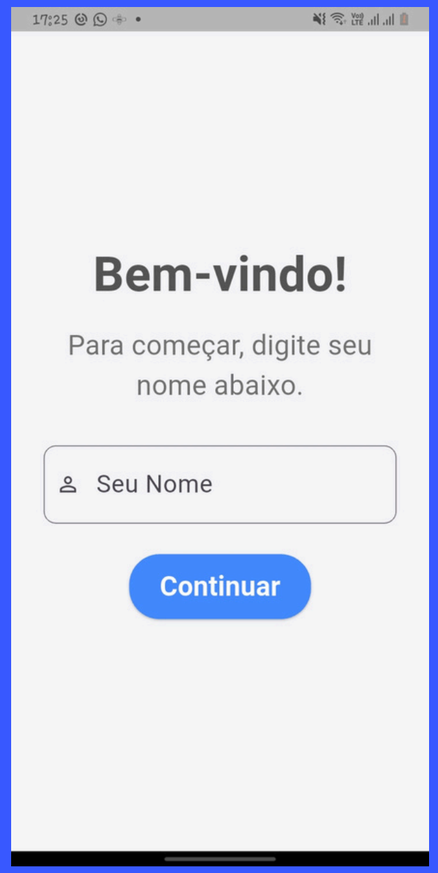
Parte Prática - ETAPA 1:

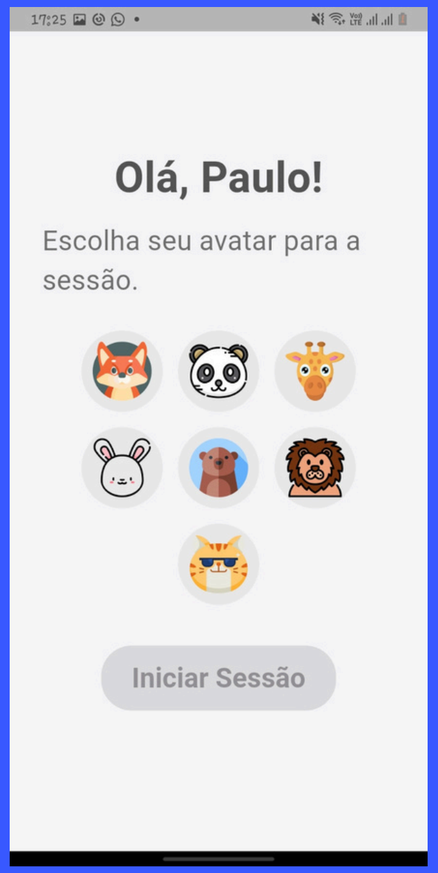
```
int motor1Pin1 = 27;
int motor1Pin2 = 26;
int enable1Pin = 14;

void setup() {
  pinMode(motor1Pin1, OUTPUT);
  pinMode(motor1Pin2, OUTPUT);
  pinMode(enable1Pin, OUTPUT);

  Serial.begin(115200);
}

void loop() {
  // Motor para frente
  Serial.println("Motor frente");
  digitalWrite(motor1Pin1, LOW);
  digitalWrite(motor1Pin2, HIGH);
  digitalWrite(enable1Pin, HIGH); // velo
  delay(2000);

  // Motor parado
  Serial.println("Motor parado");
  digitalWrite(motor1Pin1, LOW);
  digitalWrite(motor1Pin2, LOW);
  digitalWrite(enable1Pin, LOW);
  delay(1000);
```

```
  // Motor para trás
  Serial.println("Motor trás");
  digitalWrite(motor1Pin1, HIGH);
  digitalWrite(motor1Pin2, LOW);
  digitalWrite(enable1Pin, HIGH);
  delay(2000);

  // Motor parado
  Serial.println("Motor parado");
  digitalWrite(motor1Pin1, LOW);
  digitalWrite(motor1Pin2, LOW);
  digitalWrite(enable1Pin, LOW);
  delay(1000);
}
```

Telas do Aplicativo web

Telas de Cadastro





Tela de controle e métricas

Bibliografia

- Nairck. 3D Printable Jet Engine - V3 Turbofan Driver. Disponível em: https://www.thingiverse.com/thing:4743929