# Visão Geral do Projeto

Sistema de controle e monitoramento de motor elétrico via aplicativo web, desenvolvido para a disciplina Sistemas Embarcados.

## 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

# Estrutura do Projeto

## Fase 1: Desenvolvimento Paralelo (Hardware + Software)

* **Metodologias:**
  + Hardware: Cascata (Waterfall)
  + Software: SCRUM

## Fase 2: Integração Hardware-Software

## Divisão de Equipes

### Equipe Hardware

* **Membros:** Talita e Charles
* **Responsabilidades:**
  + Controle do motor elétrico
  + Implementação de sensores de rotação
  + Modificações físicas no motor (redução de atrito)
  + Sistema de alimentação por bateria
  + Configuração do módulo WiFi

### Equipe Software

* **Membros:** Paulo e Rafael
* **Responsabilidades:**
  + Desenvolvimento do aplicativo web
  + Interface de controle de velocidade
  + Sistema de visualização de métricas
  + Sistema de fila de usuários
  + Geração de QR Code

# Arquitetura do Sistema de Controle

## Como Funciona o Controle de Rotação

### Fluxo de Controle

[App Web] → [WiFi] → [Microcontrolador] → [BTS7960] → [Motor GA25-370] → [Sensor] → [Microcontrolador] → [App Web]

### Função do BTS7960 (Driver H-Bridge)

O BTS7960 atua como "amplificador de potência" entre Microcontrolador e motor:

* Microcontrolador: Fornece sinais de controle (3.3V, ~40mA)
* BTS7960: Amplifica para potência necessária (6V, até 450mA para o motor)
* Proteções integradas: Sobrecorrente, sobretensão, superaquecimento
* Controle bidirecional: Permite rotação horária e anti-horária

### Controle PWM (Modulação por Largura de Pulso)

* 0% PWM: Motor parado
* 50% PWM: Motor à meia velocidade
* 100% PWM: Motor na velocidade máxima (169 RPM)
* Frequência: Até 25kHz (padrão Arduino ~1kHz)

### Alimentação

* Bateria 12V → Regulador 6V → Motor
* Bateria 12V → Regulador 5V → BTS7960
* Bateria 12V → Regulador 3.3V → Microcontrolador

# Hardware

## Componentes Definidos

* Motor: GA25-370 (169 RPM @ 6V, corrente ~0.45A com carga)
* Driver de Motor: BTS7960 H-Bridge (43A máximo, PWM até 25kHz)
* Microcontrolador: Microcontrolador-WROOM-32 DevKit (WiFi integrado, 3.3V)
* Sensor de rotação: Sensor óptico ou magnético
* Alimentação: Bateria 12V com reguladores (6V motor, 5V BTS7960, 3.3V Microcontrolador)
* Componentes auxiliares: Reguladores de tensão, capacitores, resistores

## Especificações Técnicas

* Controle de velocidade: PWM 0-100% via BTS7960
* Direções: Horária e anti-horária (H-Bridge completo)
* Monitoramento: RPM real via Sensor, corrente via sensor integrado
* Comunicação: WiFi 802.11 b/g/n integrado no Microcontrolador
* Proteções: Sobrecorrente, sobretensão, superaquecimento (BTS7960)
* Autonomia: Dependente da bateria escolhida

## Requisitos do Microcontrolador

* **Para o BTS7960:**
  + Tensão de controle: 3.3V ~ 5V
  + PWM: Até 25kHz
  + Pinos digitais: Mínimo 4 (RPWM, LPWM, R\_EN, L\_EN)
  + Corrente de saída: Baixa (apenas sinal de controle)
* **Para o Motor GA25-370:**
  + Alimentação: 6V DC (separada do microcontrolador)
  + Controle: Via BTS7960 (não direto)
* **Para o Projeto:**
  + WiFi: Obrigatório para aplicativo web
  + PWM: Para controle de velocidade
  + ADC: Para leitura de sensores (Sensor, corrente)

## KY-003 Hall Effect Sensor

* Funcionamento: Digital (HIGH/LOW)
* Alimentação: 3.3V-5V
* Saída: Sinal digital limpo
* Instalação: Fixar próximo ao motor + ímã no eixo
* **Uso:**
  + Ímã pequeno colado no eixo do motor (ou acoplado)
  + Sensor fixo próximo ao eixo
  + Cada volta completa = 1 pulso no microcontrolador
  + RPM = (pulsos/minuto)
* **Para controle do motor:**
  + Envia dados digitais para o BTS7960 que os traduz em pulsos e envia para o motor

# Software

## Stack

* Frontend: Flutter
* Backend: Cloud Functions for Firebase
* Comunicação: Cloud Firestore (em modo de escuta/snapshot)
* Banco de Dados: Cloud Firestore
* Estilização: Widgets e Animações do Flutter

## Funcionalidades Principais

1. **Interface de Controle**
   * Velocímetro visual (similar ao de carro)
   * Controle deslizante de velocidade
   * Botões de emergência/parada
2. **Sistema de Métricas**
   * RPM atual
   * Consumo de energia estimado
   * Tempo de operação
   * Gráficos históricos
   * Eficiência energética
3. **Sistema de Usuários**
   * Registro simples (apenas nome)
   * Fila de espera
   * Tempo limite por sessão
   * Lista de nomes proibidos (regex)
4. **Conectividade**
   * Geração de QR Code automático para acessar o site
   * SSID e senha padrão configuráveis
   * Interface responsiva para mobile

## Funcionalidades Detalhadas

### Sistema de Controle de Acesso

* Acesso único: Apenas 1 usuário controlando por vez
* Fila de espera: Sistema de turnos automático
* Tempo limite: 5-10 minutos por sessão (configurável)
* QR Code: Acesso rápido via smartphone

### Visualização de Dados

* Velocímetro Digital: Interface visual atrativa
* **Métricas em Tempo Real:**
  + RPM atual
  + Potência consumida
  + Eficiência
  + Temperatura (se sensor disponível)
* Histórico: Gráficos das últimas sessões

### Sistema de Validação

* Nomes Proibidos: Lista configurável
* Regex: Validação de padrões
* Sanitização: Prevenção de caracteres especiais

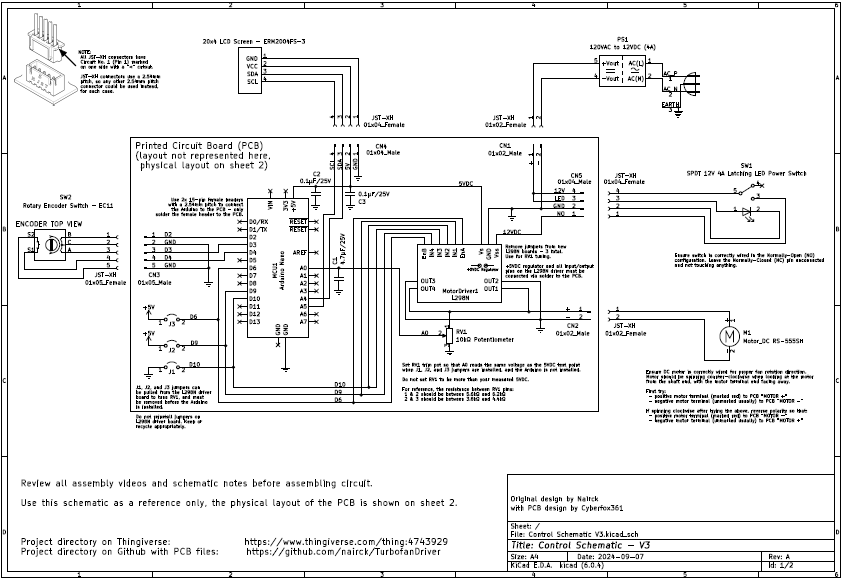
## Métricas Calculadas

Com os dados de RPM do Sensor, calcularemos:

* Velocidade angular (rad/s, RPM)
* Potência mecânica (P = τ × ω)
* Eficiência energética (Saída/Entrada × 100%)
* Consumo de corrente via sensor BTS7960
* Energia acumulada (kWh)
* Tempo de operação por sessão

Sprint 2: PARTE PRÁTICA

Primeiramente, analisou-se o projeto eletrônico disponível em <https://www.thingiverse.com/thing:4743929?utm_source=chatgpt.com>.



A partir do mesmo definiu-se os materiais, adaptando ao nosso projeto em especifico.

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)

**Cronograma adaptado:**

### SEMANA 1-2: Planejamento e Aquisição

**Semana 1: Até dia 28**

* Estudar o projeto completo no GitHub
* Fazer lista detalhada de materiais (baseada na imagem)
* Pesquisar fornecedores e preços
* Encomendar componentes eletrônicos

Atividade Realizada? Sim!

**Semana 2:**

* Aguardar chegada dos componentes
* Começar código para integração com o sistema.
* Estudar esquemático elétrico
* Preparar ambiente de desenvolvimento (Arduino IDE/PlatformIO)
* Já tentar fazer motor dc girar

### SEMANA 3-4: Eletrônica Básica

**Semana 3:**

* Montar circuito básico em protoboard
* modelo motor, sensor
* Testar ESP32 +Driver + Motor
* Implementar controle PWM básico
* Testar LCD e interface inicial

**Semana 4:**

* Adicionar encoder rotativo
* Sensor
* Implementar leitura de RPM
* Testar push button e controles
* Debug e ajustes da eletrônica

### SEMANA 5-6: Software Completo

**Semana 5:**

* Implementar interface de usuário no LCD
* Sistema de menus e configurações
* Controle de velocidade com feedback
* Algoritmo de controle PID (se necessário)

**Semana 6:**

* Implementar todas as funcionalidades
* Testes extensivos de software
* Calibração e ajustes finos
* Documentação do código

### SEMANA 7-8: Montagem e Finalização

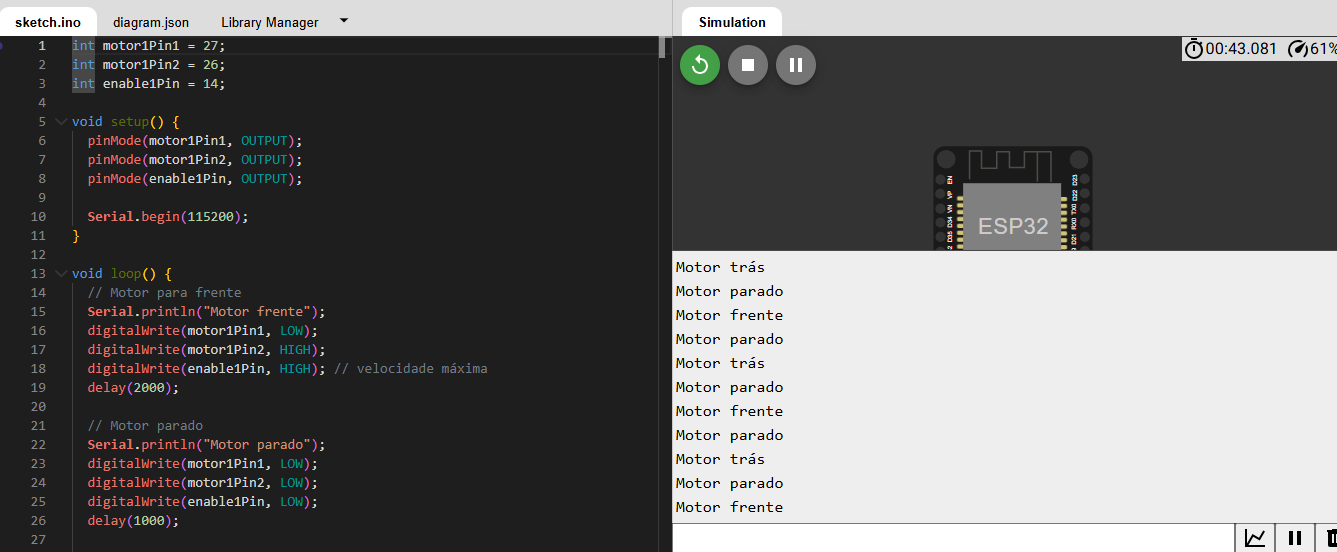
**Semana 7:**

* Soldar PCB final (se for fazer)
* Montagem mecânica completa
* Testes integrados hardware + software
* Ajustes e correções

**Semana 8:**

* Testes finais e validação
* Documentação do projeto
* Preparação da apresentação
* Buffer para imprevistos

**Simulação Controle das Rotações do Motor:**



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);

  // 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);

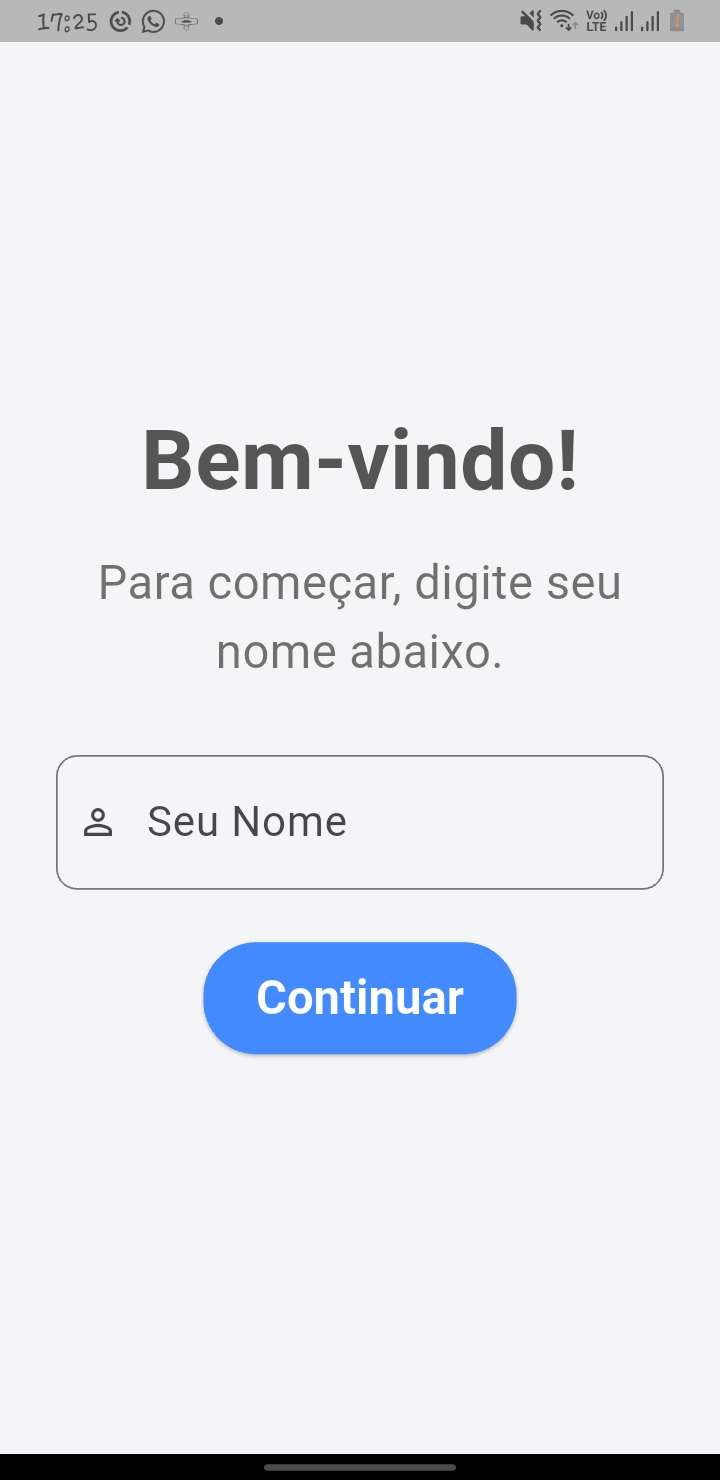
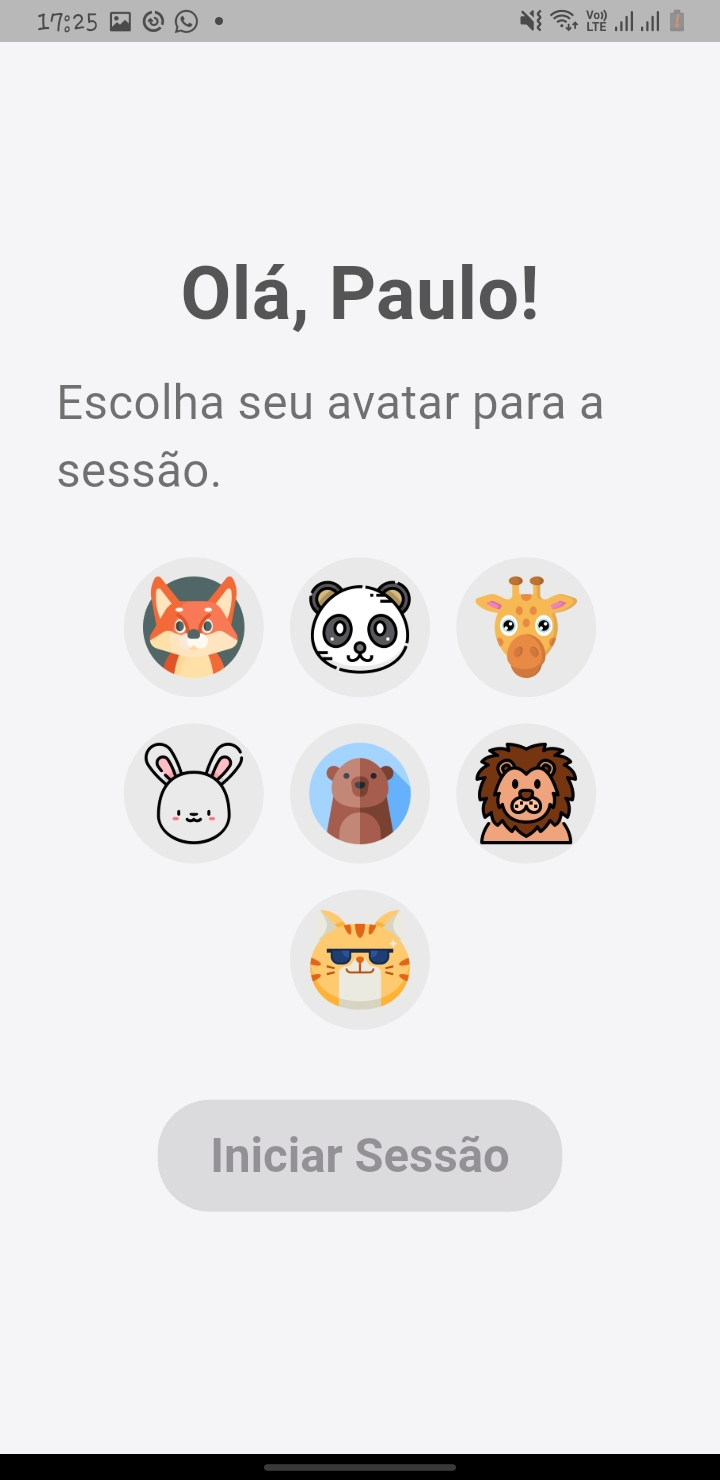
  digitalWrite(enable1Pin, LOW);

  delay(1000);

}

DESENVOLVIMENTO INICIAL DAS TELAS

**Telas de Cadastro**

**Tela de controle do velocímetro e visualização das métricas**