

# **Prototipação em Engenharia**



## **Portfólio**

**Aluno: Marco Antonio Monteiro Pedro**

<https://github.com/marcosamambaia/CamisaParaCego>

# Sumário

Acessibilidade e Independência para Pessoas Cegas: A Solução da Camisa Sensorial .....	3
Como a camisa sensorial melhora a mobilidade? .....	3
Projeto Técnico: Camisa Sensorial para Pessoas Cegas com ESP32 .....	4
Objetivo .....	4
Arquitetura do Sistema .....	4
Distribuição dos Sensores e Motores .....	4
Esquema no Protheus Schematic: .....	5
Esquema no Protheus PCB: .....	6
Funcionamento .....	7
Código-Fonte .....	7
Potenciais Melhorias .....	10

# Acessibilidade e Independência para Pessoas Cegas: A Solução da Camisa Sensorial

A vida de uma pessoa cega é repleta de desafios diários. A locomoção em ambientes urbanos pode ser complexa, exigindo atenção constante para evitar obstáculos como postes, paredes, veículos e até mesmo pedestres. A ausência de uma percepção visual limita a segurança e pode dificultar a independência. Muitas pessoas cegas dependem de bengalas, cães-guia e aplicativos de acessibilidade, mas nem sempre esses recursos são suficientes para garantir uma experiência fluida e intuitiva.

Pensando nisso, a **camisa sensorial baseada no ESP32** surge como uma solução inovadora para aumentar a autonomia das pessoas cegas. Utilizando **sensores ultrassônicos HC-SR04**, a camisa consegue detectar obstáculos próximos e alertar o usuário por meio de **motores de vibração** estrategicamente posicionados. O sistema conta com **quatro sensores**, localizados nos ombros, na frente e nas costas, proporcionando uma percepção espacial mais ampla.

## Como a camisa sensorial melhora a mobilidade?

**Detecção de obstáculos em tempo real** – Sensores identificam objetos ao redor e enviam alertas rapidamente.

**Alertas intuitivos por vibração** – Cada motor de vibração responde a um sensor específico, permitindo ao usuário entender onde o obstáculo está localizado.

**Facilidade de uso** – Basta vestir a camisa para que ela comece a auxiliar na navegação do ambiente.

**Maior segurança e independência** – Redução do risco de colisões e acidentes, permitindo que a pessoa se move com mais confiança.

Essa tecnologia pode ser integrada a outros dispositivos assistivos, criando um sistema robusto para melhorar a qualidade de vida das

pessoas cegas. Com mais desenvolvimento e testes, podemos aperfeiçoar essa inovação e torná-la acessível para um público maior.

A **camisa sensorial** representa um grande passo para a acessibilidade e autonomia, trazendo um mundo mais seguro e inclusivo para aqueles que enfrentam dificuldades diárias de locomoção.

## Projeto Técnico: Camisa Sensorial para Pessoas Cegas com ESP32

### Objetivo

Desenvolver um sistema vestível baseado em **ESP32 (NodeMCU-32S)**, utilizando **sensores ultrassônicos HC-SR04** e **motores de vibração**, para auxiliar pessoas cegas na detecção de obstáculos ao redor.

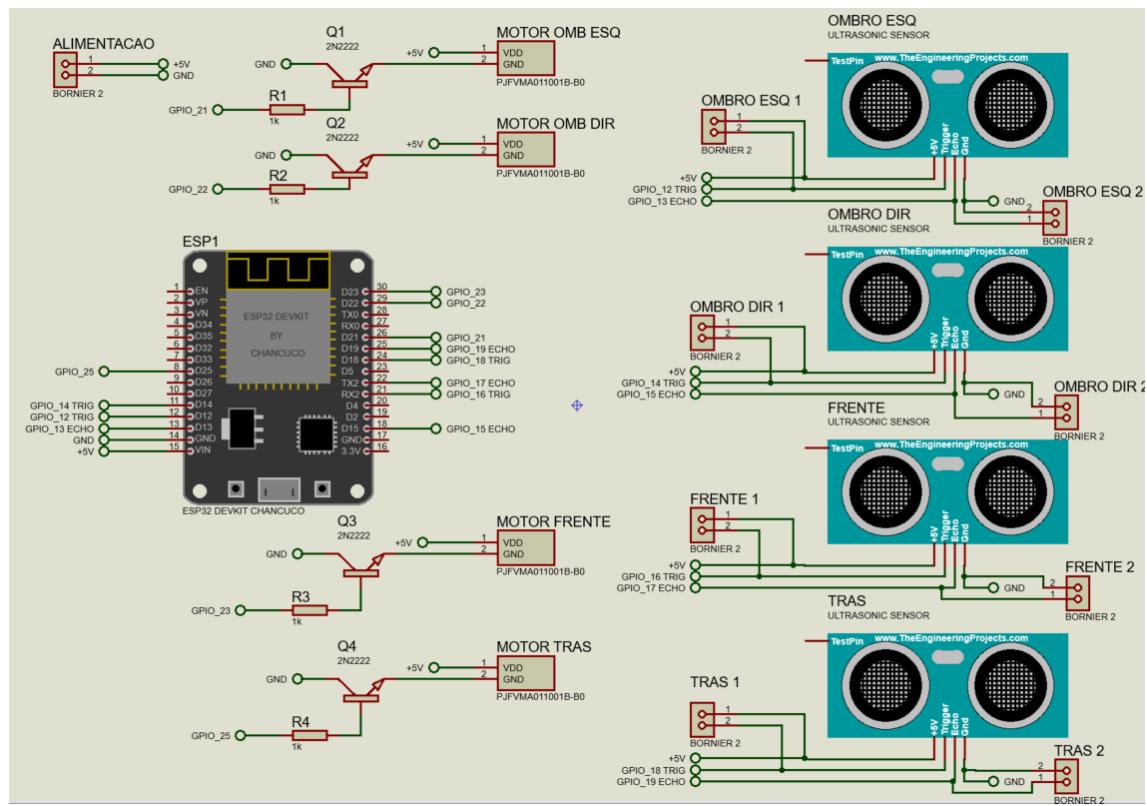
### Arquitetura do Sistema

**Placa de controle:** ESP32 (NodeMCU-32S) – Responsável pelo processamento e comunicação entre os sensores e atuadores.  
**Sensores ultrassônicos (HC-SR04)** – Mede a distância de obstáculos.  
**Motores de vibração** – Alertam o usuário sobre obstáculos próximos.  
**Fonte de alimentação** – Bateria recarregável para mobilidade.

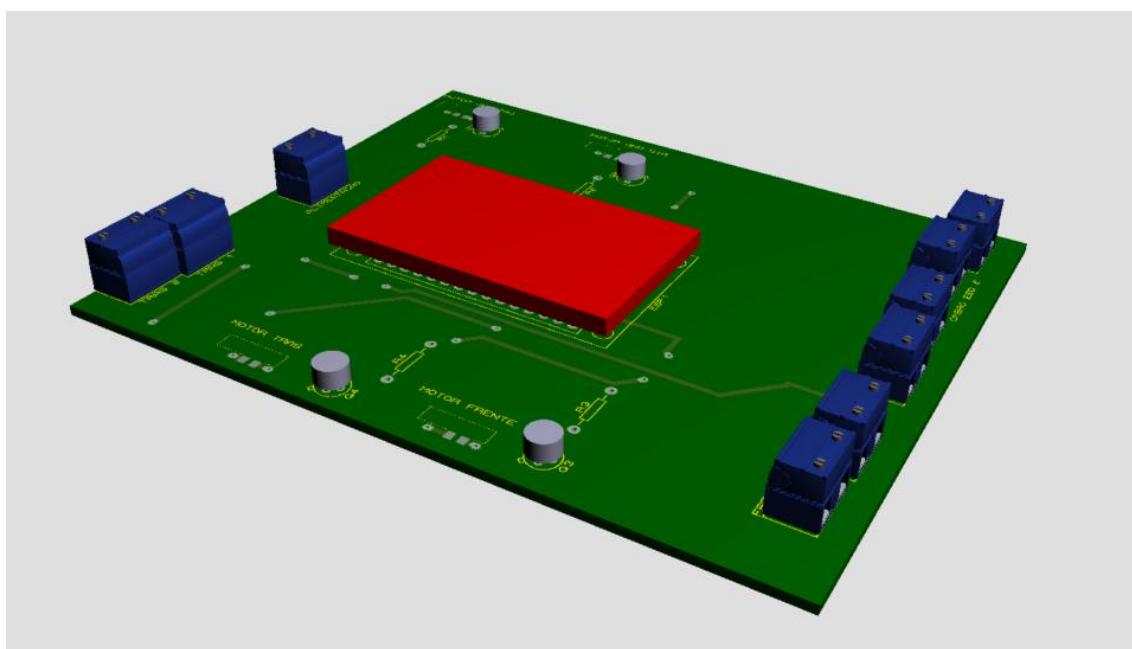
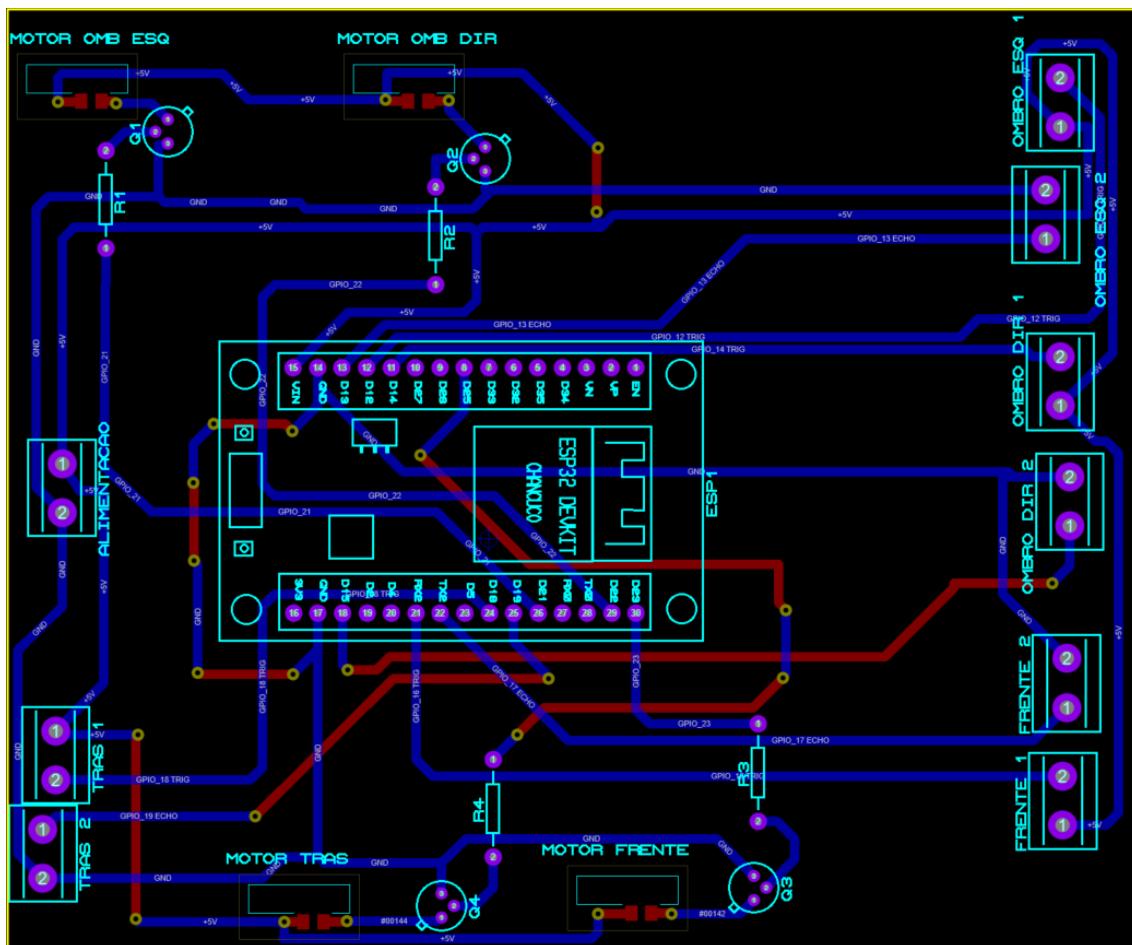
### Distribuição dos Sensores e Motores

**Sensor Ombro Esquerdo** → GPIO 12 (TRIG) / GPIO 13 (ECHO) → Motor GPIO 21  
**Sensor Ombro Direito** → GPIO 14 (TRIG) / GPIO 15 (ECHO) → Motor GPIO 22  
**Sensor Frontal** → GPIO 16 (TRIG) / GPIO 17 (ECHO) → Motor GPIO 23  
**Sensor Traseiro** → GPIO 18 (TRIG) / GPIO 19 (ECHO) → Motor GPIO 25

## Esquema no Protheus Schematic:



## Esquema no Protheus PCB:



## Funcionamento

- 1 Captura de dados** – O ESP32 ativa os sensores HC-SR04, que enviam ondas ultrassônicas e capturam o tempo de retorno do eco.
- 2 Processamento das distâncias** – O microcontrolador calcula a distância em centímetros com base na velocidade do som.
- 3 Ativação dos motores** – Se um obstáculo estiver a **menos de 50 cm**, o motor correspondente vibra para alertar o usuário.
- 4 Resposta em tempo real** – O sistema roda em loop contínuo, garantindo a atualização constante das informações.

## Código-Fonte

O código é implementado em **C++**, utilizando **PlatformIO no VS Code**. A estrutura inclui:

- Biblioteca Arduino.h** para controle do ESP32
- Função medirDistancia(trigPin, echoPin)** para calcular a distância dos sensores
- Condicionais digitalWrite(motor, HIGH/LOW)** para ativação dos motores.

Segue o código:

```
#include <Arduino.h> // Biblioteca necessária para usar funções do Arduino

// Definição dos pinos dos sensores ultrassônicos
#define TRIG_ESQUERDO 12 // Sensor no ombro esquerdo
#define ECHO_ESQUERDO 13

#define TRIG_DIREITO 14 // Sensor no ombro direito
#define ECHO_DIREITO 15
```

```
#define TRIG_FRENTES 16 // Sensor frontal
#define ECHO_FRENTES 17

#define TRIG_TRASEIROS 18 // Sensor traseiro
#define ECHO_TRASEIROS 19

// Definição dos pinos dos motores de vibração
#define MOTOR_ESQUERDO 21 // Motor do ombro esquerdo
#define MOTOR_DIREITO 22 // Motor do ombro direito
#define MOTOR_FRENTES 23 // Motor frontal
#define MOTOR_TRASEIROS 25 // Motor traseiro

void setup() {
    Serial.begin(115200); // Inicializa a comunicação serial

    // Configuração dos sensores ultrassônicos
    pinMode(TRIG_ESQUERDO, OUTPUT);
    pinMode(ECHO_ESQUERDO, INPUT);

    pinMode(TRIG_DIREITO, OUTPUT);
    pinMode(ECHO_DIREITO, INPUT);

    pinMode(TRIG_FRENTES, OUTPUT);
    pinMode(ECHO_FRENTES, INPUT);

    pinMode(TRIG_TRASEIROS, OUTPUT);
    pinMode(ECHO_TRASEIROS, INPUT);

    // Configuração dos motores de vibração
```

```

pinMode(MOTOR_ESQUERDO, OUTPUT);
pinMode(MOTOR_DIREITO, OUTPUT);
pinMode(MOTOR_FRENT, OUTPUT);
pinMode(MOTOR_TRAS, OUTPUT);

}

// Função para medir distância de um sensor ultrassônico
int medirDistancia(int trigPin, int echoPin) {
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    long duration = pulseIn(echoPin, HIGH);
    return duration * 0.034 / 2; // Retorna a distância calculada em cm
}

void loop() {
    // Medindo distância de cada sensor
    int distanciaEsquerdo = medirDistancia(TRIG_ESQUERDO, ECHO_ESQUERDO);
    int distanciaDireito = medirDistancia(TRIG_DIREITO, ECHO_DIREITO);
    int distanciaFrente = medirDistancia(TRIG_FRENT, ECHO_FRENT);
    int distanciaTras = medirDistancia(TRIG_TRAS, ECHO_TRAS);

    // Ativando os motores conforme a proximidade dos obstáculos
    digitalWrite(MOTOR_ESQUERDO, (distanciaEsquerdo < 50) ? HIGH : LOW);
    digitalWrite(MOTOR_DIREITO, (distanciaDireito < 50) ? HIGH : LOW);
    digitalWrite(MOTOR_FRENT, (distanciaFrente < 50) ? HIGH : LOW);
}

```

```
digitalWrite(MOTOR_TRAS, (distanciaTras < 50) ? HIGH : LOW);

// Pequena pausa para estabilizar as leituras
delay(100);

}
```

## Potenciais Melhorias

**Redução do consumo de energia** com otimização do código

**Integração com Bluetooth** para futuras notificações sonoras

**Aprimoramento do design** para maior conforto na vestimenta

Esse documento pode servir como base para a documentação oficial do seu projeto no **GitHub** e em futuras melhorias. Se precisar de ajustes ou mais detalhes, é só falar!