Dispositivo Vestível de Auxílio à Navegação para Deficientes Visuais com Feedback Tátil, Sensoriamento de Obstáculos e Rastreamento GPS

Autores:

- João Victor Benício
- Raul Perez
- Vinicius Macedo Escudeiro
- Bruno Chia

Instituição: Instituto Mauá de Tecnologia

Curso: Engenharia de Controle e Automação

Disciplinas:

- Microcontroladores
- Instrumentação

Professores:

- Profa. Andressa Corrente Martins
- Prof. Rodrigo França

Data: 27 de Novembro de 2025

Resumo

Este relatório descreve o desenvolvimento de um dispositivo vestível projetado para auxiliar a navegação de deficientes visuais. O sistema integra uma cinta com feedback tátil, um protótipo de radar para detecção de obstáculos e um módulo GPS para rastreamento de localização. O projeto serve como uma investigação preliminar para um futuro Trabalho de Conclusão de Curso (TCC) focado na adaptação de um jet-ski para pilotagem por deficientes visuais. O sistema é composto por dois circuitos independentes, garantindo a privacidade do usuário ao permitir a desativação do rastreamento por GPS. O primeiro circuito, baseado em um microcontrolador Raspberry Pi Pico W, gerencia um sensor de distância ultrassônico montado em um motor de passo para emular um radar, e aciona quatro motores vibratórios para fornecer feedback tátil direcional. O segundo circuito, também utilizando um Raspberry Pi Pico W, é dedicado ao módulo GPS e à comunicação Bluetooth com uma interface de monitoramento em um aplicativo móvel. O desenvolvimento abrangeu desde a concepção e teste de componentes até a montagem de um protótipo funcional e o desenvolvimento de uma placa de circuito impresso (PCB). Os resultados validam a viabilidade do conceito de feedback tátil para orientação espacial, ao mesmo tempo que expõem as limitações dos componentes de baixo custo, como o alcance do sensor ultrassônico e a precisão do motor de passo. As conclusões deste trabalho fornecem uma base sólida para futuras melhorias, incluindo a integração de um radar comercial, o aumento da resolução do feedback tátil e a adição de orientação por áudio.

---

1. Introdução

O desenvolvimento de tecnologias assistivas tem como objetivo primordial a promoção da autonomia e da qualidade de vida para pessoas com deficiência. Para indivíduos com deficiência visual, a navegação em ambientes desconhecidos representa um desafio constante, muitas vezes dependendo de auxílios tradicionais como bengalas ou cães-guia. Com o avanço da microeletrônica e dos sistemas embarcados, surgem novas possibilidades para a criação de dispositivos mais sofisticados e eficazes.

Este projeto se insere nesse contexto como um estudo introdutório e exploratório para um futuro Trabalho de Conclusão de Curso (TCC), que ambiciona desenvolver um

sistema completo para monitorar e instrumentar um jet-ski, permitindo que deficientes visuais possam pilotá-lo de forma segura através de auxílio tátil ou sonoro.

O foco do presente trabalho é a concepção e validação de três subsistemas chave: uma interface de feedback tátil utilizando motores vibratórios, um protótipo de radar para detecção de obstáculos e um sistema de rastreamento de localização via GPS com interface móvel. A integração e o teste desses componentes visam avaliar a viabilidade da abordagem e identificar os desafios técnicos para o projeto final.

---

2. Objetivos

2.1. Objetivo Geral

Desenvolver e avaliar um protótipo de dispositivo vestível para auxílio à navegação de deficientes visuais, validando os conceitos de feedback tátil para orientação espacial, sensoriamento de obstáculos com um radar de baixo custo e monitoramento de localização por GPS.

2.2. Objetivos Específicos

- **Testar a interface com motores vibratórios:** Construir uma cinta com quatro motores vibratórios e desenvolver um sistema que traduza a localização de obstáculos em feedback tátil, variando a intensidade da vibração de acordo com a proximidade.
- **Desenvolver um protótipo de radar:** Montar e programar um sistema de varredura de ambiente utilizando um sensor de distância ultrassônico acoplado a um motor de passo, emulando o funcionamento de um radar para mapear obstáculos em 360°.
- **Analisar o módulo GPS e a comunicação Bluetooth:** Implementar um circuito independente com um módulo GPS para capturar coordenadas geográficas e transmiti-las via Bluetooth para um aplicativo móvel, compreendendo as dificuldades e limitações da tecnologia, especialmente em relação à privacidade e à estabilidade do sinal.

---

3. Materiais e Métodos

3.1. Componentes Utilizados

Componente	Quantidade	Especificação	Função no Projeto
Microcontrolador	2	Raspberry Pi Pico W	Processamento central dos dois subsistemas
Sensor de Distância	1	HC-SR04	Medição de distância para o radar
Motor de Passo	1	NEMA 17 (ou similar)	Rotação do sensor ultrassônico
Driver de Motor de Passo	1	TMC2209	Controle preciso do motor de passo
Motores de Vibração	4	Com massa desbalanceada	Atuadores para feedback tátil
Driver Ponte H	2	L298N	Controle dos motores de vibração
Módulo GPS	1	Módulo GPS genérico	Aquisição de coordenadas geográficas
Módulo Bluetooth	1	Integrado ao Pico W	Comunicação com o aplicativo móvel
Placa de Circuito Impresso	1	Produzida pela Mauá	Integração dos componentes do subsistema de radar
Fonte de Alimentação	-	Baterias/Fonte externa	Alimentação dos circuitos

3.2. Metodologia de Desenvolvimento

O projeto foi executado seguindo uma metodologia incremental, dividida nas seguintes etapas:

1. **Idealização:** Definição do escopo do projeto, dos objetivos e da arquitetura geral do sistema, separando-o nos dois subsistemas principais (radar/tátil e GPS/comunicação).
2. **Escolha e Compra de Componentes:** Pesquisa e aquisição dos componentes eletrônicos necessários com base nos requisitos de funcionalidade e custo.
3. **Teste Individual de Cada Componente:** Verificação do funcionamento de cada componente de forma isolada (sensores, atuadores, microcontroladores) para garantir sua integridade e compreender suas características de operação.
4. **Montagem de Protótipo na Protoboard:** Integração dos componentes em uma protoboard para testes de circuito e lógica de programação em um ambiente de fácil modificação.
5. **Teste e Depuração de Código:** Desenvolvimento do firmware em MicroPython para os microcontroladores, com foco na interação entre sensores e atuadores, e depuração iterativa do código para corrigir falhas e otimizar o desempenho.
6. **Desenvolvimento de Placa de Circuito Impresso (PCB) e Soldagem:** Projeto do esquemático do circuito do subsistema de radar e confecção de uma PCB para uma montagem mais robusta e confiável dos componentes.

---

4. Arquitetura do Sistema

Com o objetivo de garantir a privacidade do usuário, a arquitetura do sistema foi dividida em dois circuitos completamente independentes e sem comunicação de dados entre si.

4.1. Subsistema 1: Radar e Feedback Tátil

Este subsistema é o núcleo da orientação espacial. É controlado por um Raspberry Pi Pico W e composto pelos seguintes elementos:

- **Radar Sintético:** Um sensor ultrassônico HC-SR04 é montado sobre um motor de passo. O motor realiza uma varredura de 360°, parando em intervalos para que o sensor meça a distância até o obstáculo mais próximo em uma determinada direção. O sistema registra o número de passos do motor para derivar o ângulo de detecção.

- **Lógica de Atuação:** O espaço ao redor do usuário é dividido em quatro quadrantes (0-90°, 90-180°, 180-270°, 270-360°). Ao detectar um obstáculo, o sistema identifica em qual quadrante ele se encontra e aciona o motor de vibração correspondente.
- **Feedback Tátil:** Quatro motores de vibração são posicionados em uma cinta, correspondendo à frente (12h), lateral direita (3h), costas (6h) e lateral esquerda (9h). A intensidade da vibração de cada motor é inversamente proporcional à distância medida, fornecendo uma percepção intuitiva da proximidade do obstáculo.

4.2. Subsistema 2: GPS e Comunicação Bluetooth

Este subsistema opera de forma autônoma e é responsável pelo rastreamento de localização:

- **Aquisição de Dados:** Um segundo Raspberry Pi Pico W é conectado a um módulo GPS, lendo continuamente os dados de geolocalização (latitude, longitude, altitude, número de satélites).
- **Comunicação Sem Fio:** Os dados de GPS são transmitidos via Bluetooth para um smartphone.
- **Interface de Monitoramento:** Um aplicativo simples, desenvolvido na plataforma App Inventor, recebe e exibe os dados de localização, permitindo que um acompanhante possa monitorar a posição do usuário em tempo real.

---

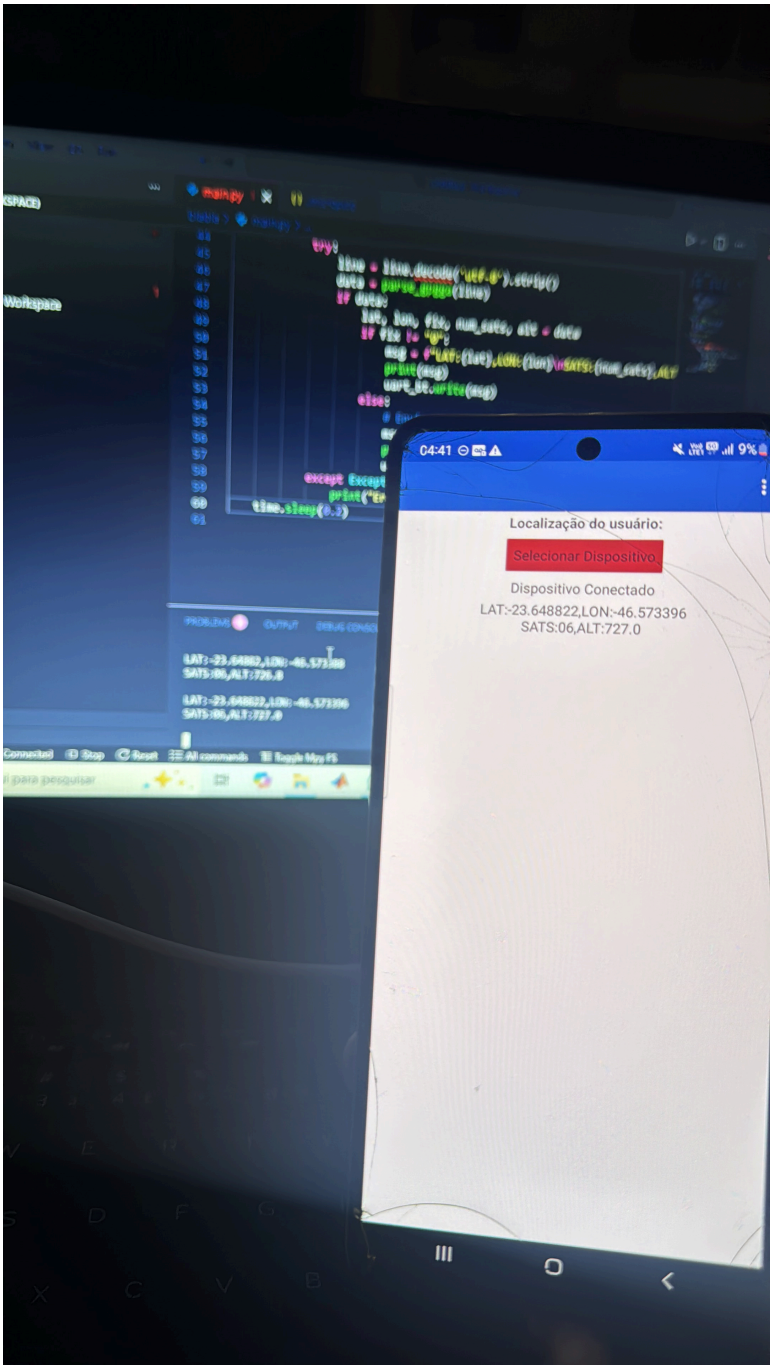
5. Resultados e Discussão

O desenvolvimento do projeto permitiu alcançar resultados funcionais, validando a prova de conceito, mas também revelou desafios importantes.

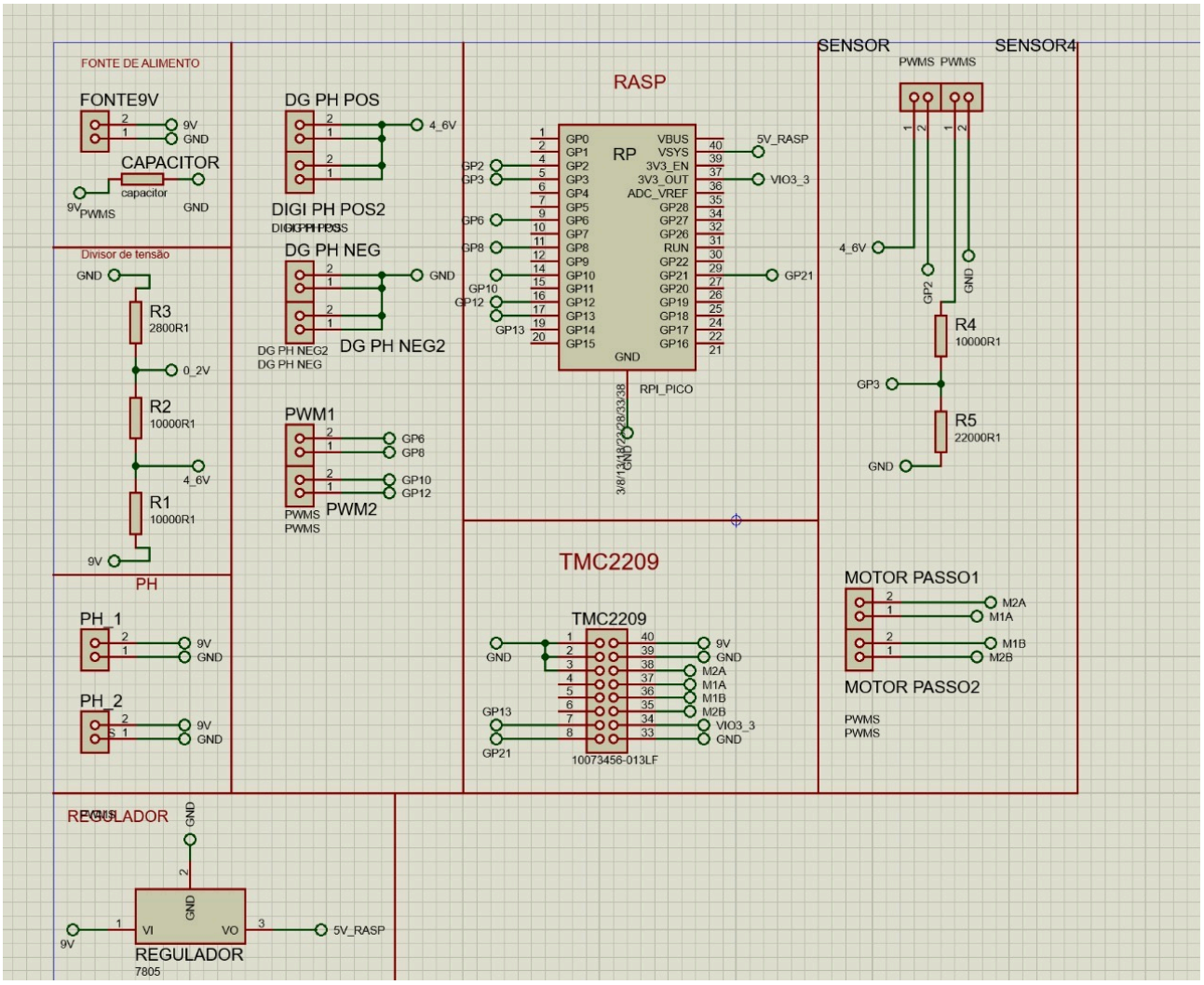
5.1. Funcionalidade do Protótipo

O sistema, especialmente durante a fase de testes em protoboard, operou conforme o esperado. O radar sintético foi capaz de realizar a varredura do ambiente, e o feedback tátil na cinta respondeu corretamente à detecção de obstáculos, com a intensidade da vibração variando de acordo com a distância. O subsistema de GPS também se

mostrou funcional, capturando coordenadas e transmitindo-as com sucesso para o aplicativo de monitoramento, como demonstrado na Figura 1.



O desenvolvimento da placa de circuito impresso (Figura 2) para o subsistema de radar representou um avanço significativo em relação à montagem em protoboard, resultando em um protótipo mais estável e compacto.



5.2. Desafios e Limitações

Diversas dificuldades foram encontradas durante o desenvolvimento:

- **Limitações do Radar Sintético:** O sensor ultrassônico HC-SR04, apesar de seu baixo custo, possui um alcance limitado e um cone de detecção relativamente amplo, o que pode levar a imprecisões. A velocidade de varredura é restrita pela necessidade de rotação do motor de passo e pelo tempo de resposta do sensor.
- **Precisão do Motor de Passo:** Foi observado que, após uso contínuo, o motor de passo pode perder passos, resultando em um erro acumulado na determinação do ângulo e, consequentemente, na localização do feedback tátil.
- **Conectividade e Sinais:** O módulo GPS apresentou dificuldade em obter sinal em ambientes fechados, uma limitação inerente à tecnologia. A comunicação via Bluetooth também se mostrou suscetível a interferências e perda de conexão com o aumento da distância entre o dispositivo e o smartphone.

- **Integração e Contatos:** Problemas de mau contato nos conectores e fios foram uma fonte recorrente de falhas durante a fase de prototipagem. A integração do software com o hardware, especialmente a calibração da resposta do sensor de distância, exigiu depuração extensiva.

---

6. Conclusão

O projeto atingiu com sucesso seus objetivos principais, demonstrando a viabilidade de um sistema vestível com feedback tátil para auxílio à navegação. A prova de conceito validou que a utilização de motores vibratórios com intensidade variável é uma forma intuitiva e eficaz de comunicar informações espaciais sobre obstáculos ao usuário.

As limitações encontradas, no entanto, são cruciais para a evolução do projeto. A montagem do radar sintético, embora funcional para validação, não se mostra robusta o suficiente para uma aplicação real devido às imprecisões do sensor ultrassônico e à perda de passos do motor. A resolução espacial do feedback, limitada a quatro quadrantes, também pode ser aprimorada para fornecer uma orientação mais precisa.

Conclui-se que, embora a abordagem seja promissora, a transição para um produto final, como o sistema de auxílio para pilotagem de jet-ski, exigirá a substituição de componentes chave por soluções de nível comercial e maior robustez.

---

7. Trabalhos Futuros

Com base nas conclusões deste trabalho, as seguintes melhorias são propostas para o desenvolvimento futuro do projeto, visando a aplicação no TCC:

- **Aumentar a Resolução Tátil:** Utilizar um número maior de motores vibratórios na cinta para fornecer uma indicação de direção mais granular e precisa.
- **Implementar Feedback Sonoro:** Adicionar um fone de ouvido ao sistema para fornecer informações complementares por áudio, como alertas de voz (“Obstáculo à direita”, “Velocidade: 10 km/h”), utilizando o conceito de áudio espacial para indicar a direção.

- **Substituir o Radar Sintético:** Incorporar um módulo de radar comercial (tecnologia de micro-ondas ou LiDAR) para obter uma detecção de obstáculos mais rápida, precisa e com maior alcance, eliminando os problemas mecânicos do motor de passo.

Apêndices

Apêndice A: Código-Fonte do Subsistema de GPS e Bluetooth

(MicroPython)

```
from machine import UART, Pin
import time

# UART do GPS (NEO-6M)
uart_gps = UART(1, baudrate=9600, tx=Pin(4), rx=Pin(5), timeout=1000)

# UART do Bluetooth (HC-05)
uart_bt = UART(0, baudrate=9600, tx=Pin(0), rx=Pin(1))


def nmea_to_decimal(coord, direction):
    if not coord:
        return None

    try:
        coord = float(coord)
    except ValueError:
        return None

    degrees = int(coord // 100)
    minutes = coord - (degrees * 100)
    decimal = degrees + minutes / 60

    if direction in ("S", "W"):
        decimal = -decimal

    return decimal


def parse_gprmc(sentence):
    parts = sentence.split(',')

    if len(parts) < 7:
        return None

    if parts[2] != 'A':  # A = válido, V = inválido
        return None

    lat = nmea_to_decimal(parts[3], parts[4])
    lon = nmea_to_decimal(parts[5], parts[6])

    if lat is None or lon is None:
        return None
```

```
    return lat, lon


def bt_send(text):
    try:
        if isinstance(text, str):
            text = text.encode()
        uart_bt.write(text)
    except Exception as e:
        print("Erro ao enviar BT:", e)


print("Iniciando GPS... Vá para perto de uma janela.")
buffer = b""

while True:

    if uart_gps.any():
        ch = uart_gps.read(1)

        if not ch:
            continue

        if ch in (b'\n', b'\r'):
            # Concluiu uma linha NMEA
            try:
                line = buffer.decode().strip()
            except:
                buffer = b""
                continue

            buffer = b""  # limpa buffer

            if line.startswith("$GPRMC") or line.startswith("$GNRMC"):
                result = parse_gprmc(line)

                if result:
                    lat, lon = result
                    msg = f"Lat: {lat:.6f}, Lon: {lon:.6f}\n"
                    print("GPS:", msg.strip())
                    bt_send(msg)
                else:
                    print("Sem fix GPS")
                    bt_send("Sem sinal GPS\n")  # <--- envia texto escolhido
```

```
        else:
            buffer += ch

            # Evita buffer infinito
            if len(buffer) > 200:
                buffer = b""

    time.sleep_ms(5)
```

Apêndice B: Código-Fonte do Subsistema de Radar e Feedback Tátil

(MicroPython)

```
// main.c
//
// Varredura com motor de passo (TMC2209) + sensor ultrassônico,
// 4 motores vibratórios controlados por PWM.
// O motor gira 360° no sentido horário, depois 360° no sentido anti-
horário,
// repetindo esse ciclo. São feitas 12 medições por volta (a cada 30°).
//
// Ajuste os parâmetros de microstepping conforme a configuração física do
TMC2209.
//

#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/pwm.h"

// ------------------------
// Configurações gerais do motor de passo
// ------------------------

// Motor de 1,8° => 200 passos cheios por volta
#define FULL_STEPS_PER_REV        850

// Fator de microstep configurado no TMC2209 (ajuste conforme jumpers /
UART)
// Exemplos típicos: 1, 2, 4, 8, 16...
#define MICROSTEPS                8

// Total de pulsos STEP necessários para 1 volta mecânica
#define STEPS_PER_REV   (FULL_STEPS_PER_REV * MICROSTEPS)

// Velocidade do motor de passo (ajuste para mais rápido ou mais devagar)
#define STEP_PULSE_US             50    // largura do pulso STEP em us
#define STEP_MIN_DELAY_US         200    // espera mínima entre passos em us

// ------------------------
// Configurações das medições
// ------------------------

// Número de medições desejadas por volta (pedido: 12 -> 360° / 12 = 30°)
#define MEASUREMENTS_PER_REV     12

// Separação angular entre medições (em graus)
```

```
#define ANGLE_BETWEEN_MEASUREMENTS  (360.0f / MEASUREMENTS_PER_REV)

// Distância máxima considerada (cm) para o mapeamento de intensidade
// Aqui começamos a "considerar obstáculo" a partir de 2,5 m.
#define MAX_DISTANCE_CM          250.0f

// Fator de suavização da intensidade (1 = sem suavização, >1 = suaviza
mais)
#define SMOOTHING_FACTOR         10

// Habilitar prints de depuração (0 = desliga, 1 = liga)
#define DEBUG                    0

// ------------------------
// Definições p/ ultrassom
// ------------------------
#define TRIGGER_PIN 2
#define ECHO_PIN    3

// ------------------------
// Definições p/ motores de vibração (PWM)
// ------------------------
#define MOTOR_A_PIN 6
#define MOTOR_B_PIN 8
#define MOTOR_C_PIN 10
#define MOTOR_D_PIN 12

// ------------------------
// Definições p/ motor de passo + TMC2209
// ------------------------
#define STEP_PIN    13
#define DIR_PIN     21
#define EN_PIN      5

// ------------------------
// Variável global de ângulo (0..360)
// ------------------------
float angulo = 0.0f;

// ------------------------
// Funções auxiliares
// ------------------------

float read_distance_cm() {
    // Disparo do pulso no TRIGGER
    gpio_put(TRIGGER_PIN, true);
```

```
    sleep_us(10);
    gpio_put(TRIGGER_PIN, false);

    // Espera o início do ECHO (subida) com timeout
    uint32_t start_time = time_us_32();
    while (!gpio_get(ECHO_PIN) && (time_us_32() - start_time < 25000)) {
        // espera até ~25 ms para subida (equivalente a ~4 m)
    }

    // Se não subiu, retorna distância máxima (sem obstáculo próximo)
    if (!gpio_get(ECHO_PIN)) {
        return MAX_DISTANCE_CM;
    }

    uint32_t echo_start = time_us_32();

    // Espera o fim do ECHO (descida) com timeout
    while (gpio_get(ECHO_PIN) && (time_us_32() - echo_start < 25000)) {
        // espera até ~25 ms de pulso
    }

    uint32_t echo_end = time_us_32();

    // Cálculo da distância em cm (velocidade do som ~ 343 m/s)
    float distance = (echo_end > echo_start)
                     ? (echo_end - echo_start) / 2.0f / 29.1f
                     : MAX_DISTANCE_CM;

    if (distance > MAX_DISTANCE_CM) distance = MAX_DISTANCE_CM;
    if (distance < 0.0f)            distance = 0.0f;

    return distance;
}

void set_motor_pwm(uint gpio, uint16_t level) {
    uint slice   = pwm_gpio_to_slice_num(gpio);
    uint channel = pwm_gpio_to_channel(gpio);
    pwm_set_chan_level(slice, channel, level);
}

void init_motor_pwm(uint gpio) {
    gpio_set_function(gpio, GPIO_FUNC_PWM);
    uint slice = pwm_gpio_to_slice_num(gpio);
    pwm_set_wrap(slice, 1000);        // resolução do PWM (0..1000)
    pwm_set_enabled(slice, true);
}
```

```
// Dá 1 passo no motor de passo (1 micro-passo, conforme config do TMC2209)
void step_once(void) {
    gpio_put(STEP_PIN, 1);
    sleep_us(STEP_PULSE_US);
    gpio_put(STEP_PIN, 0);
    sleep_us(STEP_MIN_DELAY_US);
}

// Inicializa pinos do TMC2209 em modo STEP/DIR
void init_stepper_pins(void) {
    gpio_init(STEP_PIN);
    gpio_set_dir(STEP_PIN, GPIO_OUT);
    gpio_put(STEP_PIN, 0);

    gpio_init(DIR_PIN);
    gpio_set_dir(DIR_PIN, GPIO_OUT);
    // Vamos começar girando no sentido horário (valor lógico arbitrário)
    gpio_put(DIR_PIN, 1);

    gpio_init(EN_PIN);
    gpio_set_dir(EN_PIN, GPIO_OUT);
    // EN normalmente é ativo em nível baixo: 0 = habilita, 1 = desabilita
    gpio_put(EN_PIN, 0); // habilita o driver
}

int main() {
    stdio_init_all();

    // ------------------------
    // Inicializa ultrassom
    // ------------------------
    gpio_init(TRIGGER_PIN);
    gpio_set_dir(TRIGGER_PIN, GPIO_OUT);
    gpio_put(TRIGGER_PIN, false);

    gpio_init(ECHO_PIN);
    gpio_set_dir(ECHO_PIN, GPIO_IN);
    gpio_pull_down(ECHO_PIN);

    // ------------------------
    // Inicializa motores de vibração (PWM)
    // ------------------------
    init_motor_pwm(MOTOR_A_PIN);
    init_motor_pwm(MOTOR_B_PIN);
    init_motor_pwm(MOTOR_C_PIN);
```

```
    init_motor_pwm(MOTOR_D_PIN);

    // ------------------------
    // Inicializa motor de passo (TMC2209)
    // ------------------------
    init_stepper_pins();

    // Posição mecânica atual em passos (0..STEPS_PER_REV-1)
    int pos_step = 0;

    // Contador de passos dentro da volta atual
    int steps_in_rev = 0;

    // Direção atual: true = horário, false = anti-horário
    bool dir_cw = true;

    // Controle de medições por volta (baseado em passos)
    float steps_between_measures = (float)STEPS_PER_REV /
(float)MEASUREMENTS_PER_REV;
    float next_measure_step = 0.0f;
    int measurements_this_rev = 0;

    // Intensidade suavizada
    uint16_t last_intensity = 0;

    // Distância atual (mantida entre medições para debug se quiser)
    float distancia = MAX_DISTANCE_CM;

    while (true) {
        // ------------------------
        // Salva ângulo anterior
        // ------------------------
        float angulo_anterior = angulo;

        // ------------------------
        // Dá 1 micro-passo no motor de passo
        // ------------------------
        step_once();

        // Atualiza posição mecânica em função da direção
        if (dir_cw) {
            pos_step++;
            if (pos_step >= STEPS_PER_REV) {
                pos_step = 0;
            }
        } else {
```

```
    if (pos_step == 0) {
        pos_step = STEPS_PER_REV - 1;
    } else {
        pos_step--;
    }
}

// Atualiza contagem de passos na volta atual
steps_in_rev++;

// Converte posição em ângulo (0..360)
angulo = (pos_step * 360.0f) / (float)STEPS_PER_REV;
if (angulo >= 360.0f) {
    angulo -= 360.0f;
}

// ------------------------
// Verifica se é hora de medir (12 vezes por volta)
// ------------------------
bool do_measure = false;
if ((float)steps_in_rev >= next_measure_step &&
    measurements_this_rev < MEASUREMENTS_PER_REV) {

    do_measure = true;
    measurements_this_rev++;
    next_measure_step += steps_between_measures;
}

if (do_measure) {
    // ------------------------
    // Mede distância
    // ------------------------
    distancia = read_distance_cm();

    // Calcula intensidade de vibração (0..1000)
    float raw_intensity =
        1000.0f * (MAX_DISTANCE_CM - distancia) / MAX_DISTANCE_CM;

    if (raw_intensity < 0.0f)   raw_intensity = 0.0f;
    if (raw_intensity > 1000.0f) raw_intensity = 1000.0f;

    uint16_t intensidade = (uint16_t)raw_intensity;

    // Suaviza a intensidade
    intensidade = (uint16_t)(
        (last_intensity * (SMOOTHING_FACTOR - 1) + intensidade)
```

```
            / SMOOTHING_FACTOR
        );
        last_intensity = intensidade;

        // ------------------------
        // Lógica de ângulo -> motor A/B/C/D
        // ------------------------
        uint16_t A = 0, B = 0, C = 0, D = 0;

        if (angulo >= 0.0f && angulo < 90.0f) {
            A = intensidade;
        } else if (angulo >= 90.0f && angulo < 180.0f) {
            B = intensidade;
        } else if (angulo >= 180.0f && angulo < 270.0f) {
            C = intensidade;
        } else if (angulo >= 270.0f && angulo < 360.0f) {
            D = intensidade;
        }

        // Aplica PWM nos motores de vibração
        set_motor_pwm(MOTOR_A_PIN, A);
        set_motor_pwm(MOTOR_B_PIN, B);
        set_motor_pwm(MOTOR_C_PIN, C);
        set_motor_pwm(MOTOR_D_PIN, D);

        // Depuração opcional
    #if DEBUG
        printf("Dir: %s, passo=%d/%d, ang=%.2f, dist=%.2f cm, int=%u\n",
               dir_cw ? "CW" : "CCW",
               steps_in_rev, STEPS_PER_REV,
               angulo, distancia, intensidade);
    #endif

        // Pequena pausa opcional após medição (se o sensor precisar)
        sleep_ms(2);
    }

    // ------------------------
    // Verifica fim da volta
    // ------------------------
    if (steps_in_rev >= STEPS_PER_REV) {
        // Reseta contadores da volta
        steps_in_rev = 0;
        measurements_this_rev = 0;
        next_measure_step = 0.0f;
```

```
            // Inverte a direção:
            // se estava horário (CW), passa para anti-horário (CCW) e vice-
versa
            dir_cw = !dir_cw;
            gpio_put(DIR_PIN, dir_cw ? 1 : 0);

        #if DEBUG
            printf("Mudando direção para: %s\n", dir_cw ? "CW" : "CCW");
        #endif
        }
    }

    return 0;
}
```