Sistema de Controle Inteligente para Prensa para Raspberry Pi Pico W

Discente

Caio A. da C. Andrade

24 de fevereiro de 2025.

Conteúdo

1 Escopo do Projeto **3**
- 1.1 Apresentação do Projeto . . . 3
- 1.2 Título do Projeto . . . 3
- 1.3 Objetivos do Projeto . . . 3
- 1.4 Descrição do Funcionamento . . . 3
- 1.5 Justificativa . . . 3

2 Especificação de Hardware **4**
- 2.1 Diagrama em bloco . . . 4
- 2.2 Funções dos blocos . . . 4
- 2.3 Configurações dos blocos . . . 5
- 2.4 Comantos e registros utilizados . . . 5
- 2.5 Descrição de pinagem . . . 5
- 2.6 Circuito Completo do Hardware . . . 6

3 Especificação de Firmware **7**
- 3.1 Blocos funcionais . . . 7
- 3.2 Definição de variáveis . . . 7
- 3.3 Fluxograma . . . 8
- 3.4 Inicialização . . . 8
- 3.5 Configurações dos registros . . . 8
- 3.6 Estrutura e formato dos dados . . . 9
- 3.7 Protocolo de comunicação . . . 10
- 3.8 Pacote de dados . . . 10

4 Execução do projeto **10**
- 4.1 Metodologia . . . 10
- 4.2 Testes de validação . . . 11
- 4.3 Discussão dos resultados . . . 11

5 Referências **11**

A Fluxograma de software **13**

1 Escopo do Projeto

1.1 Apresentação do Projeto

O projeto consiste no desenvolvimento de um sistema de controle para uma prensa, permitindo o acionamento tanto no modo manual quanto no modo automático. O sistema utiliza uma Raspberry Pi Pico W integrada a um BitDogLab, proporcionando flexibilidade e precisão no controle da prensa.

1.2 Título do Projeto

Sistema de Controle Inteligente para Prensa.

1.3 Objetivos do Projeto

- Desenvolver um sistema de acionamento para uma prensa utilizando um microcontrolador.
- Implementar dois modos de operação: automático e manual.
- Utilizar uma matriz de LEDs WS2812B 5050 e buzzers para contagem de pausas.
- Utilizar um LED controlado por PWM para representar a posição da prensa.
- Integrar um display OLED SSD1306 para exibição de informações sobre o funcionamento.
- Implementar botões para controle eficiente do processo.

1.4 Descrição do Funcionamento

- Modo Automático: O ciclo é iniciado pressionando o botão A, acionando a prensa (representada por um LED azul com PWM). Ao atingir a posição máxima, o sensor de fim de curso (botão B) ativa uma contagem regressiva de 3 segundos para retornar à posição inicial. Após um intervalo de 5 segundos, o ciclo se reinicia.
- Modo Manual: O usuário pode ativar o controle manual pressionando o botão do joystick, permitindo o ajuste da posição da prensa movendo o joystick para cima (aumenta a posição da prensa) ou para baixo (diminui a posição da prensa).
- As informações do sistema são exibidas no display OLED SSD1306, fornecendo feedback em tempo real ao operador da posição da prensa, quais modos estão ativos no determinado momento, a leitura de um sensor de temperatura e uma contagem de ciclos concluídos (CC).

1.5 Justificativa

Este projeto é relevante pois automatiza o controle de uma prensa, reduzindo a necessidade de intervenção humana e melhorando a eficiência do processo. Além disso, o modo manual permite ajustes finos quando necessário, tornando o sistema mais versátil e adaptável a diferentes aplicações industriais. Durante pesquisas correlacionadas, não foram encontrados sistemas embarcados utilizando a Raspberry Pi Pico W com software para controle de prensa, mas, correlacionados, encontramos projetos para controles de LEDs e servomotores por PWM, como podemos ver em "Controlar um servo com Raspberry Pi Pico W e PWM"e "Raspberry Pi Pico: PWM Fading an LED (MicroPython)". Ainda sim, há uma limitação quanto linguagem de programação utilizada que costuma ser diferente dos demais projetos vistos na internet, comumente utilizando Micro Python. Além disso, o projeto conta com a implementação do display OLED para visualização das informações em tempo real. Desta maneira, não foram

encontrados sistemas que assimilassem todas as funcionalidades aqui utilizadas em um só programa.

2 Especificação de Hardware

2.1 Diagrama em bloco

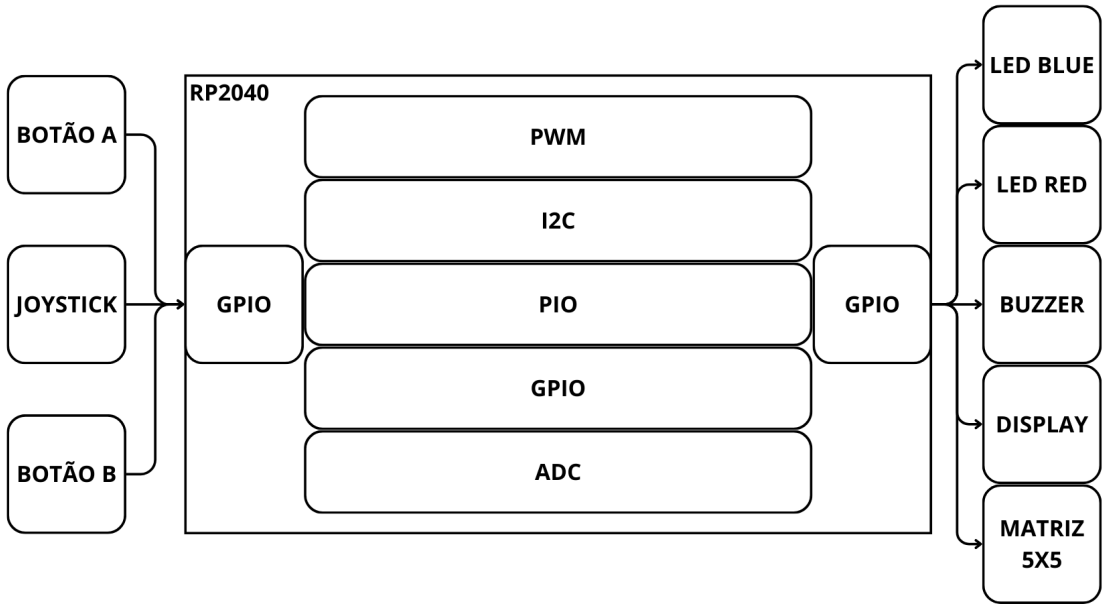

Figura 1: Diagrama em bloco do hardware.
Fonte: De autoria própria.

2.2 Funções dos blocos

1. RP2040: Os dados de entrada e saída serão processados com os respectivos protocolos definidos
2. LED RGB 5050: Cor azul representa a posição da prensa através da intensidade de luz e cor vermelha representa acionamento do modo manual.
3. Botão A: Acionamento do modo automático.
4. Botão B: Sensor de fim de curso para retorno da prensa no modo automático.
5. Joystick: Seu botão ativa o acionamento manual e o movimento do eixo Y controla a posição da prensa.
6. Buzzers: Sinal sonoro para auxilio da representação da contagem regressiva.
7. Display OLED SSD1306: Exibição de posição da prensa, estado dos modos de execução, sensor de temperatura e contagem de ciclos concluídos.
8. Matriz RGB WS2812B: Relógio contador de segundos.

2.3 Configurações dos blocos

1. RP2040: Processamento de dados.
2. LED RGB 5050: LED RED como saída em sinal baixo/alto e LED BLUE como PWM.
3. Botão A: entrada com resistor em pull up.
4. Botão B: entrada com resistor em pull up.
5. Joystick: seu botão é configurado como entrada com resistor em pull up o potenciômetro de eixo Y como entrada analógica. posição da prensa.
6. Buzzers: saída em sinal baixo/alto
7. Display OLED SSD1306: saída através de protocolo I2C.
8. Matriz RGB WS2812B: saída através de protocolo PIO.

2.4 Comantos e registros utilizados

- PWM (Pulse Width Modulation): Configuração da largura de pulso para controlar a intensidade do LED azul.
- GPIO (General Purpose Input/Output): Configuração das portas digitais para os botões, LED vermelho e buzzers.
- I2C (Inter-Integrated Circuit): Comunicação com o display SSD1306 para exibição de mensagens.
- PIO (Programmable I/O): Comunicação com a matriz RGB WS2812B.

2.5 Descrição de pinagem

- LED Azul: GPIO 12.
- LED Vermelho: GPIO 13.
- Botão A: GPIO 05.
- Botão B: GPIO 06.
- Joystick (Eixo Y): GPIO 27.
- Joystick (Botão): GPIO 22.
- Buzzer: GPIO 21.
- Display SSD1306: GPIO 14 (SDA) e GPIO 15 (SCL).
- Matriz RGB WS2812B: GPIO 07.

2.6 Circuito Completo do Hardware

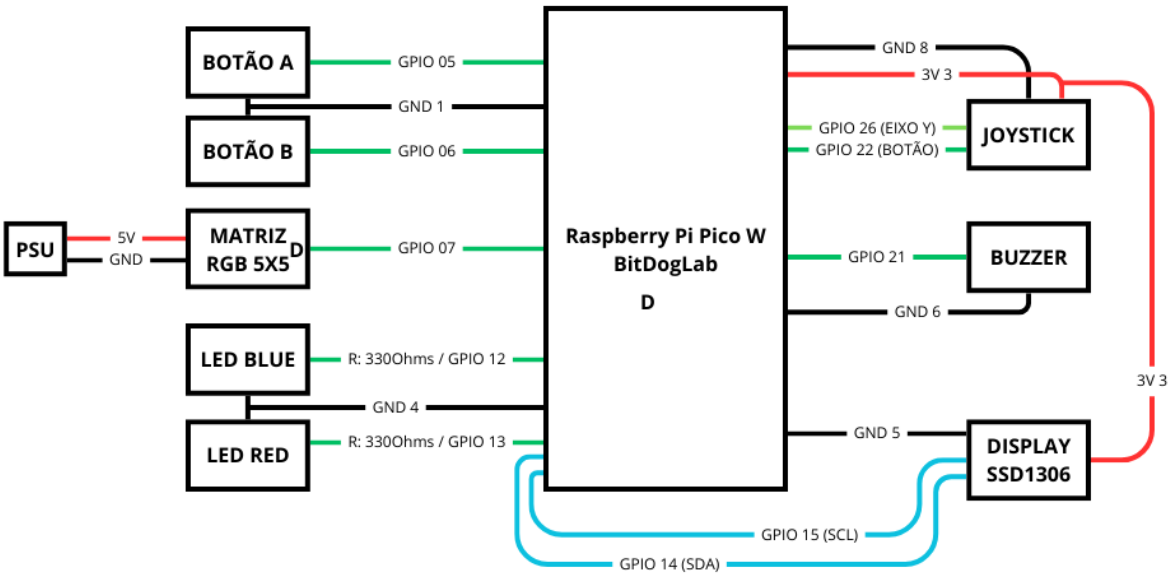

Figura 2: Circuito de hardware
Fonte: De autoria própria.

3 Especificação de Firmware

3.1 Blocos funcionais

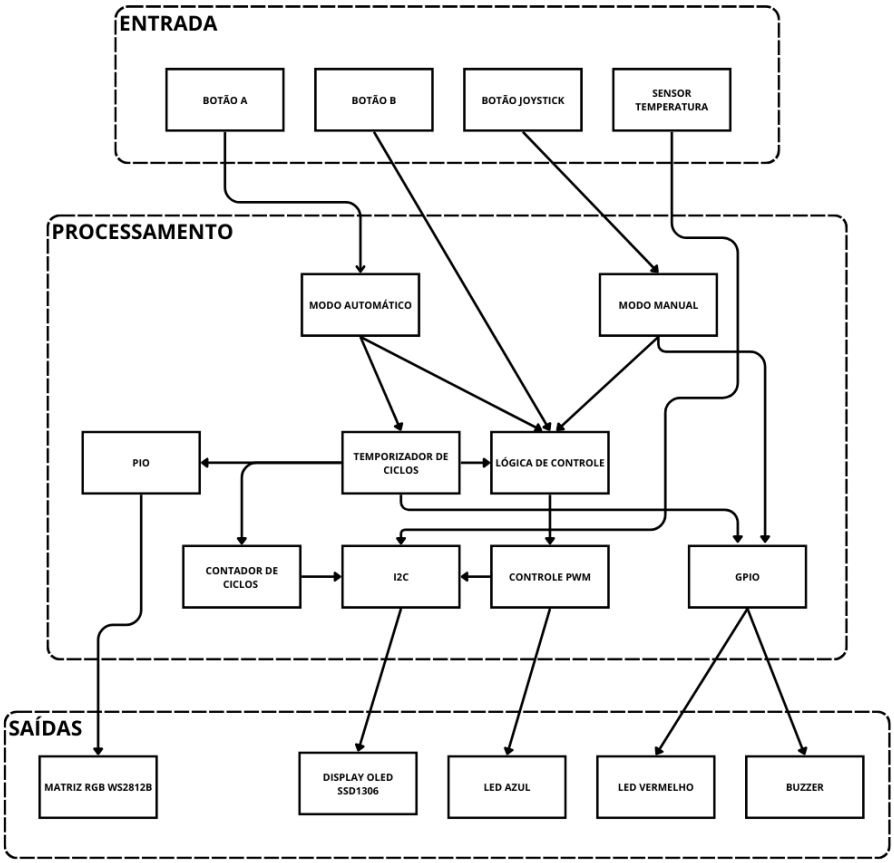

Figura 3: Diagrama de blocos de firmware.
Fonte: De autoria própria.

3.2 Definição de variáveis

- **Botões:**
  - `DEBOUNCE_DELAY` - Tempo de debounce para acionamento de botões (500 ms)
- **ADC:**
  - `vry_value` - Valor lido do ADC para o movimento do eixo Y
  - `ledy` - Valor do ADC convertido para PWM
- **PWM:**
  - `WRAP_PERIOD` - Valor do período de contagem do PWM (24999)
  - `PWM_DIVISER` - Valor do divisor de clock do PWM (100.0)
  - `pausa` - Tempo de prensagem no cíclo automático (3000 ms)
  - `pausa_f` - Pausa após recudo de prensa no cíclo automático (5000 ms)
  - `step` - Passo do movimento automático (500)
  - `stepm` - Passo do movimento manual (100)
  - `pos_prensa` - Posição da prensa

- **Matriz de LEDs:**
  - `r, g, b` - Componentes de cor para controle da matriz de LEDs
- **Display OLED:**
  - `str` - Armazena a string da posição da prensa
  - `str1` - Armazena a string da temperatura
  - `str2` - Armazena a string dos ciclos completos
  - `cont` - Contador de ciclos completos
- **Status do Sistema:**
  - `estado` - Indica se o modo automático está ativo
  - `sensorFC` - Indica se o sensor de fim de curso está pressionado
  - `manual` - Indica se o modo manual está ativo
  - `opc` - Controla a etapa do ciclo automático

3.3 Fluxograma

Verificar Anexo A.

3.4 Inicialização

A inicialização do software ocorre na função `main()`, onde as seguintes etapas são realizadas:

1. **Inicialização da Biblioteca PICO**: `stdio_init_all()` é chamada para configurar a comunicação serial padrão.
2. **Configuração do ADC**: `adc_init()` inicializa o conversor analógico-digital.
3. **Configuração dos GPIOs**: A função `init_gpios()` é chamada para configurar os pinos utilizados, incluindo botões e LEDs.
4. **Inicialização do Display**: `init_display()` configura a interface I2C e inicializa o display OLED SSD1306.
5. **Configuração do PWM**: Configuração do PWM para controlar a intensidade do LED azul (representando a prensa) e outras variáveis necessárias para o funcionamento do sistema.
6. **Configuração de Interrupções**: Configura interrupções para os botões A, B e joystick, chamando a função `gpio_set_irq_enabled_with_callback()`.
7. **Inicialização do PIO**: O PIO (Programmable Input/Output) é configurado para controlar a matriz de LEDs, usando `pio_matrix_program_init()`.

3.5 Configurações dos registros

As funções de configuração dos registros são realizadas em várias partes do código:

1. **PWM**: A função `gpio_set_function()` define o pino do LED azul como PWM. A função `pwm_set_clkdiv()` configura o divisor de clock do PWM e `pwm_set_wrap()` define o período de wrap do PWM.

```
gpio_set_function(led_b, GPIO_FUNC_PWM); // GPIO como PWM
pwm_set_clkdiv(slice, PWM_DIVISER); // Divisor de clock do PWM
pwm_set_wrap(slice, WRAP_PERIOD); // WRAP
```

Código 1: Configuração de PWM.

2. **ADC**: O pino do eixo Y do joystick é configurado como entrada analógica usando `adc_gpio_init()`, e o ADC é selecionado com `adc_select_input()`.

```
adc_select_input(0);
uint16_t vry_value = adc_read();
```

Código 2: Configuração de ADC.

3. **Display SSD1306**: A configuração do I2C é realizada na função `init_display()`, que define os pinos SDA e SCL, a frequência de trabalho e inicializa o display.

```
i2c_init(I2C_PORT, 400 * 1000); // Inicializa  o do I2C utilizando 400Khz
gpio_set_function(I2C_SDA, GPIO_FUNC_I2C); // Define GPIO como I2C
gpio_set_function(I2C_SCL, GPIO_FUNC_I2C); // Define GPIO como I2C
gpio_pull_up(I2C_SDA); // Define GPIO SDA como pullup
gpio_pull_up(I2C_SCL); // Define GPIO SCL como pullup
ssd1306_init(&ssd, WIDTH, HEIGHT, false, endereco, I2C_PORT); // Inicializa o display
ssd1306_config(&ssd); // Configura o display
```

Código 3: Configurações de display.

3.6 Estrutura e formato dos dados

Os dados e variáveis são definidos antes da inicialização do programa, visto que, a qualquer momento o desenvolvedor pode modificar parâmetros sem necessariamente realizar modificações diretamente na `main`.

```
//============BOTOES============
#define btn_a 5 // Botao A
#define btn_b 6 // Botao B
#define btn_j 22 // Botao de joystick
#define DEBOUNCE_DELAY 500 // Debounce para acionamento de botoes
volatile int ultima_interrup = 0; // Para armazenar o ultimo tempo de interrupcao

//============SAIDAS============
#define led_b 12 // LED azul
#define led_r 13 // LED vermelho
#define buzzer 21 // Buzzer

//============ADC============
#define VRY_PIN 26 // GPIO DE EIXO Y
float ledy; // ADC para PWM

//============PWM============
#define WRAP_PERIOD 24999 // Valor do WRAP
#define PWM_DIVISER 100.0 // Valor do divisor de clock do PWM
#define pausa 3000 // Tempo de prensagem
#define pausa_f 5000 // Pausa de final curso
uint16_t step = 500; // Passo de movimento automatico
uint16_t stepm = 100; // Passo de movimento manual
int pos_prensa = 0; // Posicao de prensa

//============MATRIZ============
#define NUM_LEDS 25 // Numero de LEDs
#define MATRIZ_PIN 7 // GPIO

//============DISPLAY============
#define I2C_PORT i2c1
#define I2C_SDA 14 // GPIO SDA DO DISPLAY
```

```c
#define I2C_SCL 15 // GPIO SCL DO DISPLAY
#define endereco 0x3C // ENDERECO DO DISPLAY
ssd1306_t ssd; // Inicializa a estrutura do display
char str[10]; // Armazena a string da posicao da prensa
char str1[10]; // Armazena a string da temperatura
char str2[10]; // Armazena a string de ciclos completos
int cont; // Contador de ciclos completos (CC)

//============STATUS============
volatile bool estado = false; // Status do modo automatico
volatile bool sensorFC = false; // Status de sensor de fim de curso
volatile bool manual = false; // Status de modo manual
volatile int opc; // Etapa do modo automatico
```

Código 4: Variáveis e constantes.

3.7 Protocolo de comunicação

O sistema utiliza dois protocolos principais para comunicação com periféricos:

1. I2C: Utilizado para comunicação com o display OLED. O protocolo I2C opera a 400 kHz, com os pinos SDA e SCL configurados como GPIOs.

```c
i2c_init(I2C_PORT, 400 * 1000);
gpio_set_function(I2C_SDA, GPIO_FUNC_I2C);
gpio_set_function(I2C_SCL, GPIO_FUNC_I2C);
```

2. PIO: Utilizado para comunicação com a matriz de LEDs. O PIO é programado para enviar dados de controle para os LEDs.

```c
PIO pio = pio1;
bool ok;
uint16_t i;
uint32_t valor_led;
double r, g, b;

ok = set_sys_clock_khz(128000, false);

uint sm = pio_claim_unused_sm(pio, true);
uint offset = pio_add_program(pio, &pio_matrix_program);
pio_matrix_program_init(pio, sm, offset, MATRIZ_PIN);
```

3.8 Pacote de dados

Para a implementação do sistema, foram utilizadas as seguintes bibliotecas e arquivos:

1. `font.h` - Biblioteca que contém as definições de fontes utilizadas no display.
2. `ssd1306.c` - Implementação dos scripts de controle do display SSD1306.
3. `ssd1306.h` - Cabeçalho da biblioteca SSD1306, que declara as funções e estruturas utilizadas.
4. `matrix.h` - Biblioteca que contém as funções para realizar desenhos da matriz WS2812B.

4 Execução do projeto

4.1 Metodologia

Iniciou-se o projeto com uma pesquisa sobre as tecnologias disponíveis e os componentes necessários para a construção do sistema. Para isso, foi feita uma revisão do conteúdo abordado

através do curso de capacitação EmbarcaTech. Foram estudados conceitos relacionados à programação em C para microcontroladores, comunicação I2C para o display SSD1306, e a implementação de matrizes de LEDs utilizando a tecnologia PIO da Raspberry Pi Pico W.

Levando em consideração que a placa foi pré-definida obrigatoriamente, apenas foi necessário um estudo para escolher quais componentes disponíveis melhor se adequariam ao projeto, sendo possivel tornar a interface simples e objetiva. Os componentes adicionais incluídos foram:

1. Display SSD1306, conectado via I2C, para exibição de informações.
2. Matriz de LEDs 5x5, gerenciada por PIO, e buzzer para feedback visual.
3. Botões e LEDs conectados aos GPIOs para interação com o usuário.

As funcionalidades do software foram definidas com base nos requisitos do projeto. O sistema foi projetado para controlar uma prensa, permitindo operação manual e automática. Dessa maneira, através de botões é possível controlar seus estados de funcionamento. Com a matriz RGB 5x5 e com o auxílio do buzzer é possível acompanhar em tempo real as funções de temporização presentes no programa. Além disso, com o auxílio do display OLED é possivel acompanhar de maneira objetiva todo o comportamento do programa.

A IDE escolhida para o desenvolvimento foi o Visual Studio Code. Inicializamos a IDE e instalamos as extensões necessárias, como a extensão de C/C++ para suporte a programação na linguagem C, e extensões adicionais para suporte ao desenvolvimento na Raspberry Pi Pico.

O desenvolvimento do software foi realizado por meio da escrita de código em C. As bibliotecas para o display (font.h, ssd1306.c, ssd1306.h) e para a matriz (matrix.h) foram integradas ao projeto. Além disso, utilizou-se as bibliotecas específicas para as Raspberry Pi Pico W a fim de integrar todas as funcionalidades possíveis.

Após a programação, foi realizada uma fase de depuração para identificar e corrigir possíveis erros. Testes foram conduzidos para verificar o funcionamento dos botões, a exibição no display e o controle da matriz de LEDs. Durante essa fase, foram utilizados métodos de depuração, como a impressão de mensagens para verificar o funcionamento do programa.

4.2 Testes de validação

O primeiro teste de validação foi feito através de uma simulação virtual para verificar o pleno funcionamento do código. Para isso, foi utilizada a plataforma WokiWi, onde foi incluído o esquema elétrico juntamente com o código e pacote de dados necessários. Desta forma foi possível atestar o pleno funcionamento do software antes de pôr em risco a integridade física de um hardware.

Em segundo, através da placa BitDogLab, pode-se aplicar o protótipo do projeto, assim sendo possível identificar erros de operação, como sobreposição de funções.

4.3 Discussão dos resultados

O protótipo do projeto foi concluído de maneira satisfatória. Todas as interações propostas foram aplicadas e o programa apresenta funcionamento estável. Apesar disso, em momentos raros, foi notado que o sensor de fim de curso se aciona sozinho não esperando por sua ativação manual. Tal comportamento não ocorreu de maneira padronizada, assim, levantando a hipótese de possível problema no componente físico do botão.

5 Referências

1. Random Nerd Tutorials. "Raspberry Pi Pico PWM with MicroPython."Disponível em: https://randomnerdtutorials.com/raspberry-pi-pico-pwm-micropython/. Acesso em: 18 de fevereiro de 2025.

2. Picockpit. "Como controlar servo com Raspberry Pi Pico W e modulação por largura de pulso (PWM)."Disponível em: https://picockpit.com/raspberry-pi/pt/how-to-control-servo-with-raspberry-pi-pico-w-and-pulse-width-modulation-pwm/. Acesso em: Acesso em: 18 de fevereiro de 2025.

3. Embarcatech. "Embarcatech: Treinamento em Sistemas Embarcados."Disponível em: https://embarcatech.com/.

A Fluxograma de software

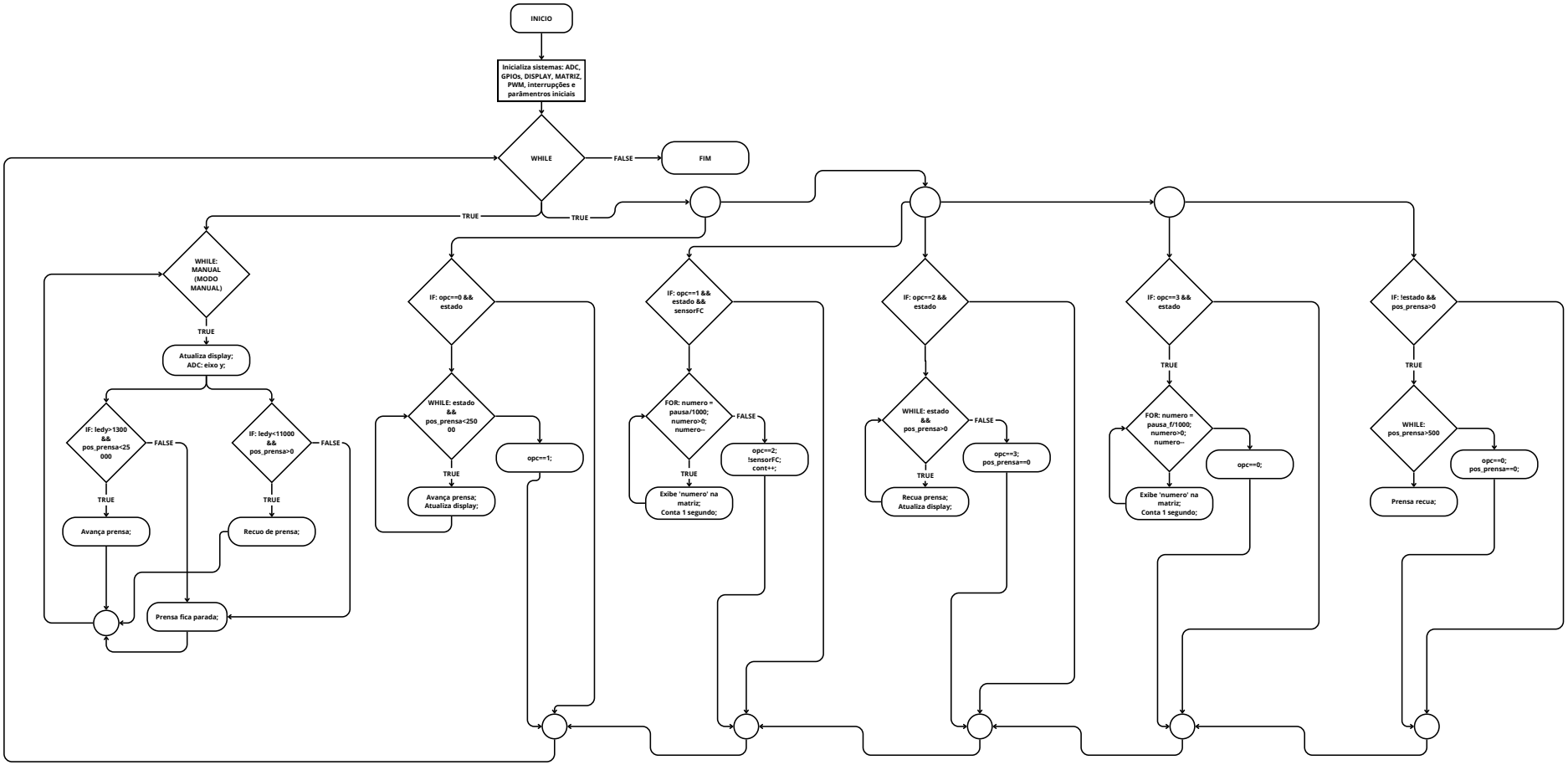

Figura 4: Fluxograma do Controle da Prensa.pdf
Fonte: De autoria própria.