RESIDÊNCIA TECNOLÓGICA EM SISTEMAS EMBARCADOS:
Formação Básica em Software Embarcado

Projeto Final: Transmissor de código Morse usando a placa de desenvolvimento BitDogLab

GABRIEL FRANÇA OLIVEIRA VIANA
TIC370100619

Ilhéus/BA
2025

Introdução

O presente trabalho é o resultado final dos conhecimentos obtidos e assimilados no curso Formação Básica em Software Embarcado coordenado pela Softex, o mesmo se trata da criação de um software para a placa de desenvolvimento BitDogLab, com o objetivo de permitir o usuário inserir uma mensagem e a mesma ser transmitida em código Morse através de um sinal luminoso.

Para a realização deste projeto foi necessário o conhecimento a respeito da placa BitDogLab e seus componentes incluindo os periféricos de entrada (botões e joystick) e saída (leds, display ssd1306 e matriz de leds), microcontrolador RP2040 equipado ao modulo raspberry PICO w e seus periféricos como GPIOs, canais ADC, recursos de comunicação UART, I2C e PWM.

A programação e testes foram feitos usando a própria placa fornecida em colaboração pela CEPEDI (centro de pesquisa e desenvolvimento tecnológico em informática e eletroeletrônica de ilhéus), IFs (Institutos federais) da região nordeste, o Instituto HARDWARE BR e Softex. O código foi desenvolvido através da IDE da Microsoft, VScode com as extensões apropriadas para o uso da linguagem C, simulação do hardware via Wokwi e desenvolvimento via Pico SDK.

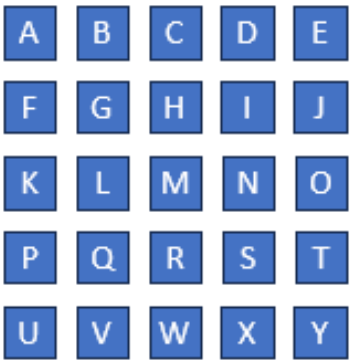
Link para o github com os arquivos do projeto: https://github.com/GabrielFOV/tarefa_final.git

Link para o vídeo no google drive: https://drive.google.com/file/d/1aDVzoxcc5_RiZP4DJe5ijouO7tqMuRZT/view?usp=sharing

Funcionamento do software

Ao iniciar o hardware a matriz led 5x5 é acionada com todos os leds ligados em azul, esse é primeiro painel e representa a os caracteres do alfabeto, um dos leds acende em branco, é um indicativo de que o carácter correspondente está selecionado.

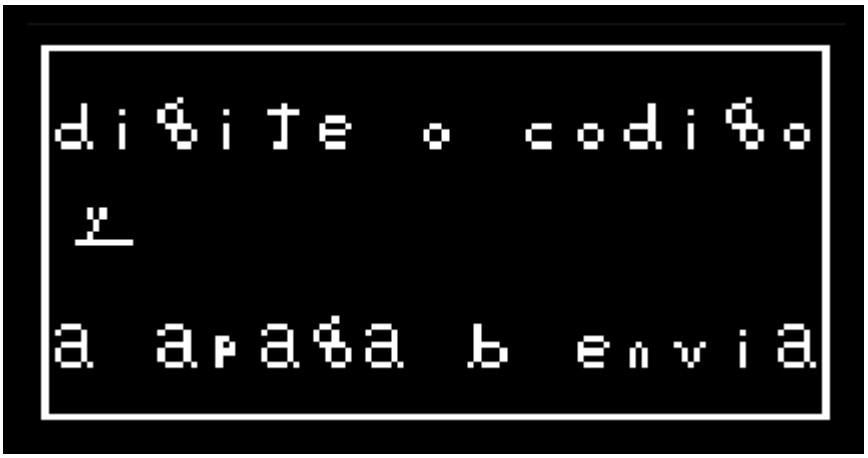
Figura 1: Matriz de led, painel 1 de caracteres



Fonte: autor, 2025.

Ao mover o joystick o led branco muda de posição, selecionando um novo caractere. O caractere selecionado no momento é indicado no display ssd1306. Abaixo é apresentada a imagem da simulação programada usando a extensão do wokwi no IDE vscode.

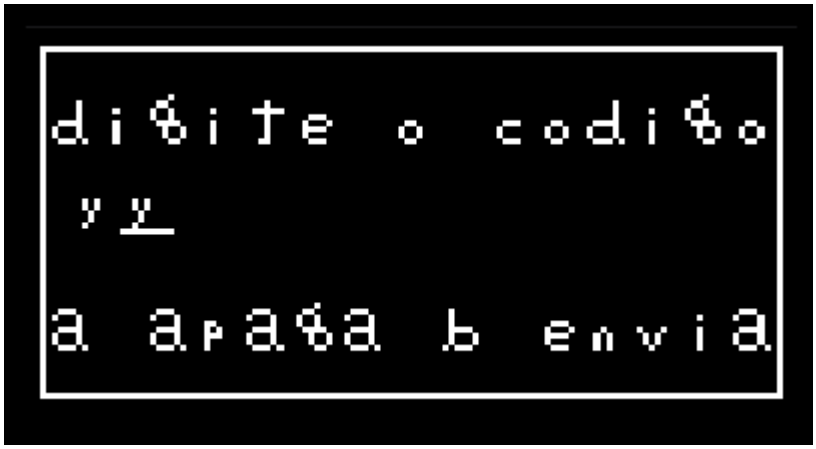
Figura 2: Display inicial apresentado



Fonte: simulador Wokwi.

A linha sob o caractere indica que ele está selecionado, mas não foi inserido no código. Pressionar o botão do joystick insere o caractere na string de texto e o apresenta na tela.

Figura 3: Display inicial com a inserção da letra y



Fonte: simulador Wokwi.

Pressionar o botão A da placa de desenvolvimento apagará o último caractere inserido, o botão B inicia a transmissão da mensagem na forma de código Morse através do acionamento e desligamento de um led seguindo a cadência de transmissão padrão.

O led inicialmente pisca em azul indicando o início da transmissão, em seguida o led pisca em branco e um led vermelho indica o final da transmissão.

Figura 4: Display durante a transmissão do código

Fonte: simulador Wokwi.

No arquivo principal (TarefaFinal.c) é possível habilitar a transmissão via UART da mensagem na forma de código Morse e o texto inserido, adaptando a forma escrita com o caractere + para representar a separação entre um caractere e outro.

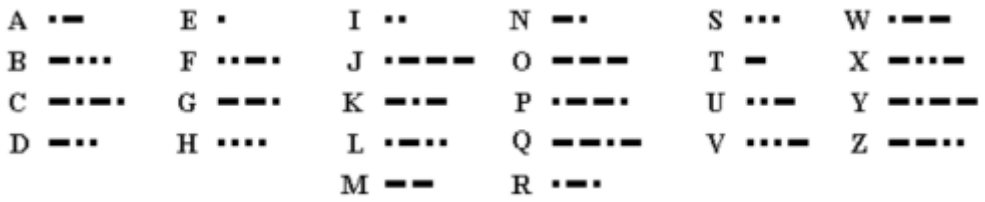
É importante ressaltar que existe uma diferença no funcionamento do código, do simulador e da placa, o eixo x é invertido no simulador, para que funcione corretamente na placa.

Código Morse

O código Morse foi inventado por Samuel Finley Breese Morse, devido a criação do telegrafo elétrico em 1838. “A nomenclatura utilizada pelo código baseia-se em pontos e traços, que correspondem a sinais elétricos. O ponto faz o papel de unidade e tem cerca de 1/25 segundos de duração. Por sua vez, o sinal elétrico relativo ao traço equivale a três pontos.” (Zebendo; Nobre; 2019).

Abaixo é apresentado um resumo da correspondência do código.

Figura 5: Caracteres alfanuméricos em código Morse



Algarismo	Sinal	Pontuação	Sinal
1	•—————	Ponto	••••••
2	••———	Ponto e vírgula	—•—•—•
3	•••——	Vírgula	•—•—•—
4	••••—	Dois pontos	———•••
5	•••••	Interrogação	••——••
6	—••••	Exclamação	——••—
7	——•••	Apóstrofe	•————•—
8	———••	Traço de união	—••••—
9	————•	Aspas	•—••—•
0	—————	Parêntesis	—•——•—
		Alínea	•—•—••
		Sublinhado	••——•—
		Duplo traço (=)	—•••—

Fonte: Zebendo; Nobre; 2019.

Na época em que foi criado, criou-se uma revolução nas telecomunicações. Este meio de comunicação foi fortemente utilizado por marinheiros no século XIX.

Atualmente, muitos radioamadores ainda o utilizam, pois é eficiente em condições de sinal fraco. Alguns faróis marítimos ainda o utilizam para indicar sua localização. Em situações de desastre onde a comunicação convencional é impossível ele se torna muito útil, sendo reconhecido internacionalmente o sinal

de socorro SOS (···– – –···), sendo também muito utilizado em jogos e desafios de decodificação.

Revisão Bibliográfica

Com base nos trabalhos relacionados encontrados, o intuito principal é proporcionar a oportunidade de praticar os conceitos de programação e aplicação de conhecimentos de hardware, passa a ser, portanto, uma forma mais interativa e lúdica de se trabalhar com dispositivos eletrônicos.

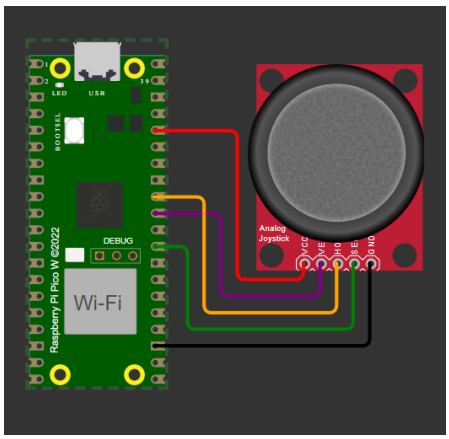
Caetano; Filho e Moreira (2020) possuem um trabalho publicado na Revista Brasileira de Ensino de Ciência e Tecnologia, onde constroem um sistema transmissor e um sistema receptor de código Morse em sinais luminosos, usando Arduino, para a pratica da disciplina Prática de Ensino de Física II – FIS262.

Não há muitos trabalhos acadêmicos que tratem diretamente das formas de trazer o código Morse ao ambiente de programação e implementá-lo na linguagem C, mas em fóruns como o Stack overflow, geeksforGeeks e clubdohardware é possível encontrar várias discussões relacionadas e porções codificadas.

Componentes de Hardware

Dentre os componentes de hardware utilizados estão o joystick, que na verdade se trata de dois potenciômetros alinhados perpendicularmente entre si, centrados sobre um push button. O joystck tem por característica, funcionar como uma entrada analógica de sinal, logo é necessário usar o ADC para ler e interpretar sua entrada.

Figura 6: conexão do joystick ao modulo raspberry PICO w



Fonte: simulador Wokwi.

Na figura 6 são demonstradas as conexões do joystick a placa raspberry PICO w, o fio vermelho é a entrada de tensão de 3,3V, o fio preto é o terra/neutro conectado ao GND.5, o fio verde a entrada para o push button ligada a porta GP22, o fio amarelo a entrada do eixo vertical GP26 e o fio roxo a entrada do eixo horizontal na porta GP27.

Os principais comandos e funções utilizados foram:

Inicializar o ADC:

-adc_init();

-adc_gpio_init(VRX_PIN);

-adc_gpio_init(VRY_PIN);

Inicializar o botão

-gpio_init(SW_PIN);

-gpio_set_dir(SW_PIN, GPIO_IN);

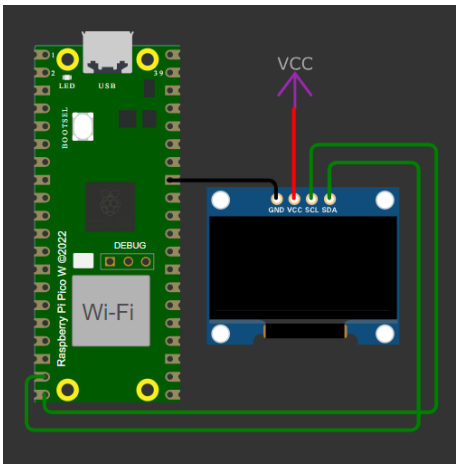
-gpio_pull_up(SW_PIN);

Interrupção:

-gpio_set_irq_enabled(SW_PIN, GPIO_IRQ_EDGE_FALL, true);

Outro componente usado é o display SSD1306 que usa um protocolo de comunicação I2C.

Figura 7: conexão do display SSD1306 ao modulo raspberry PICO w



Fonte: simulador Wokwi.

A linha de dados (SDA) está conectada a porta GP14, a linha de clock SCL para a sincronização está conectada na porta GP15, o neutro/terra a GND.7 e a entrada de tensão á representação do simulador, no caso de uma aplicação real haveria uma protoboard ou outro tipo de circuito entre a entrada do display e modulo raspberry, mas por praticidade de representação no simulador, usa-se o símbolo VCC para a representação.

Os principais comandos e funções utilizados foram:

Inicializar a comunicação I2C:

- i2c_init(I2C_PORT, 400 * 1000);

gpio_set_function(I2C_SDA, GPIO_FUNC_I2C);

gpio_set_function(I2C_SCL, GPIO_FUNC_I2C);

gpio_pull_up(I2C_SDA);

gpio_pull_up(I2C_SCL);

Inicializar e configurar o display:

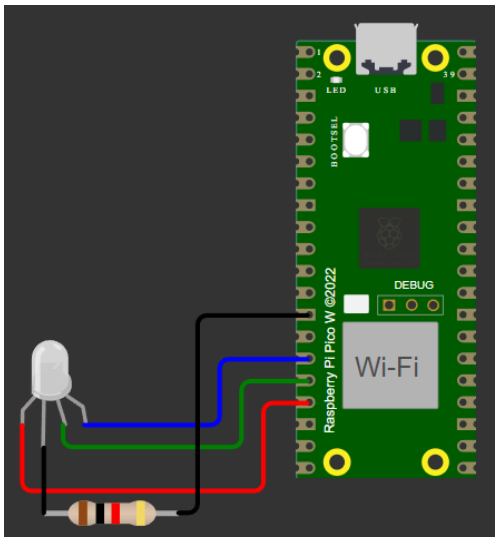
-void ssd1306_init(ssd1306_t *ssd, uint8_t width, uint8_t height, bool external_vcc, uint8_t address, i2c_inst_t *i2c) {

```
  ssd->width = width;
  ssd->height = height;
  ssd->pages = height / 8U;
  ssd->address = address;
  ssd->i2c_port = i2c;
  ssd->bufsize = ssd->pages * ssd->width + 1;
```

```
  ssd->ram_buffer = calloc(ssd->bufsize, sizeof(uint8_t));
  ssd->ram_buffer[0] = 0x40; .}
- void ssd1306_config(ssd1306_t *ssd) {
  ssd1306_command(ssd, SET_DISP | 0x00);
  ssd1306_command(ssd, SET_MEM_ADDR);
  ssd1306_command(ssd, 0x01);
  ssd1306_command(ssd, SET_DISP_START_LINE | 0x00);
  ssd1306_command(ssd, SET_SEG_REMAP | 0x01);
  ssd1306_command(ssd, SET_MUX_RATIO);
  ssd1306_command(ssd, HEIGHT - 1);
  ssd1306_command(ssd, SET_COM_OUT_DIR | 0x08);
  ssd1306_command(ssd, SET_DISP_OFFSET);
  ssd1306_command(ssd, 0x00);
  ssd1306_command(ssd, SET_CONTRAST);
  ssd1306_command(ssd, 0xFF);
  ssd1306_command(ssd, SET_DISP | 0x01); }
-ssd1306_send_data(&ssd);
```

O led RGB é utilizado como forma de transmitir o sinal em código Morse.

Figura 8: conexão do led RGB ao modulo raspberry PICO w



Fonte: simulador Wokwi.

Como pode ser observado pela coloração dos fios, a componente vermelha do led está ligada a porta GP13, o verde, a porta GP12, o azul, a porta GP11 e o terra/neutro ao GND.3.

Os principais comandos e funções utilizados foram:

Inicializar os led:

-gpio_init(ledB_pin);

-gpio_set_dir(ledB_pin, GPIO_OUT);

-gpio_init(ledA_pin);

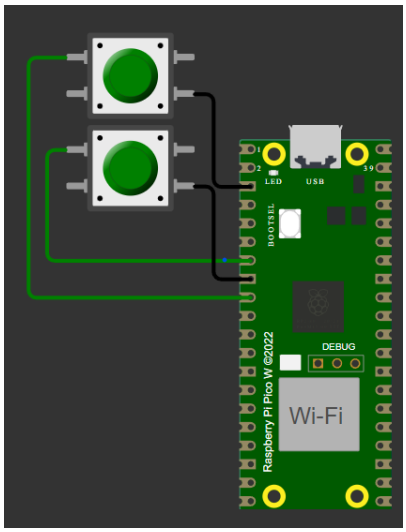
-gpio_set_dir(ledA_pin, GPIO_OUT);

-gpio_init(ledC_pin);

-gpio_set_dir(ledC_pin, GPIO_OUT);

Dois push buttons também são utilizados.

Figura 9: conexão dos botões ao modulo raspberry PICO w



Fonte: simulador Wokwi.

O push button A (inferior) está ligado a porta GP5 e GND.2, o push button B está ligado a porta GP6 e GND.1.

Os principais comandos e funções utilizados foram:

Inicializar os botões e o pull up:

-gpio_init(button_A);

-gpio_set_dir(button_A, GPIO_IN);

-gpio_pull_up(button_A);

-gpio_init(button_B);

-gpio_set_dir(button_B, GPIO_IN);

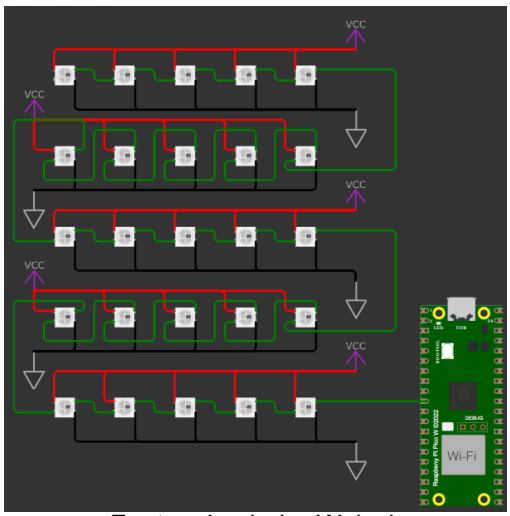
-gpio_pull_up(button_B);

Interrupções:

-gpio_set_irq_enabled_with_callback(button_A, GPIO_IRQ_EDGE_FALL, true, &gpio_irq_handlerA);
-gpio_set_irq_enabled(button_B, GPIO_IRQ_EDGE_FALL, true);

Por último, a matriz led 5x5 usa vários leds NeoPixel WS2812, 25 no total, ligados em serie.

Figura 10: conexão da matriz led ao modulo raspberry PICO w



Fonte: simulador Wokwi.

A entrada de tensão e o neutro de cada led estão ligados a representações, tal qual o display pelo mesmo motivo supracitado. A entrada de dados de um led está ligada a saída do anterior, e a comunicação com a placa se dá pela porta GP7.

Os principais comandos e funções utilizados foram:

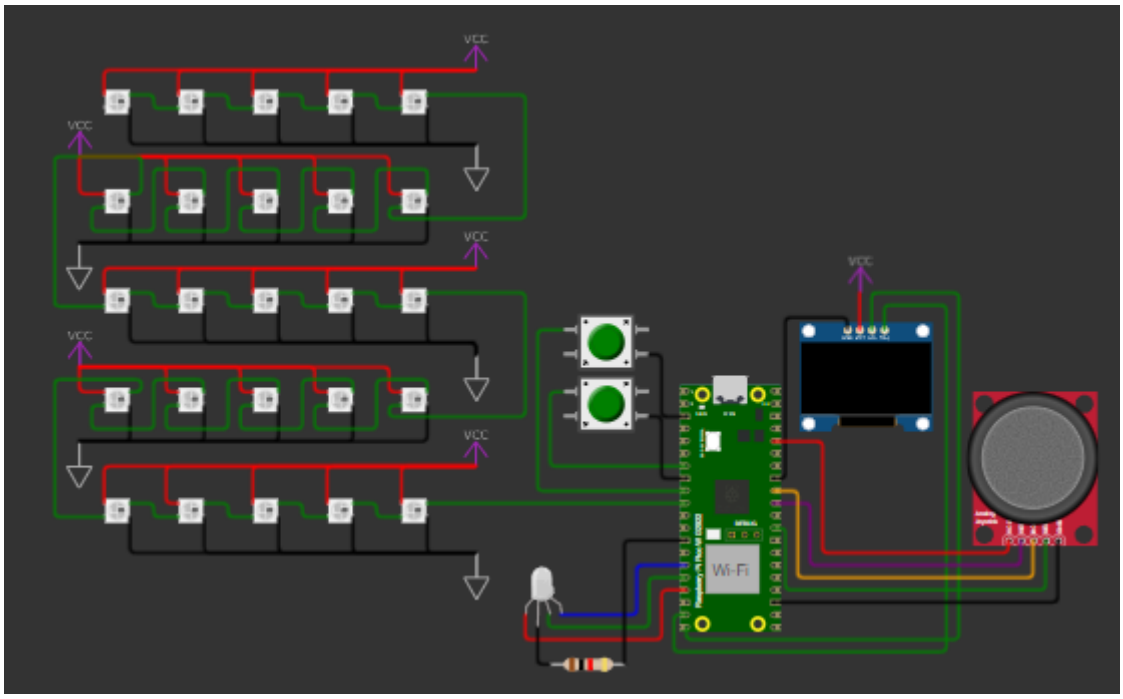
-npInit(LED_PIN);
-npClear();
-npWrite();

Tabela 1: descrição da pinagem

Porta (GPIO)	Função	Componente
0	UART TX	Comunicação Serial
1	UART RX	Comunicação Serial
5	Botão A	Entrada Digital
6	Botão B	Entrada Digital
7	Matriz de LEDs	WS2818B (PIO)
11	LED Verde	Saída Digital
12	LED Azul	Saída Digital
13	LED Vermelho	Saída Digital
14	I2C SDA	Display OLED
15	I2C SCL	Display OLED
22	Botão Joystick	Entrada Digital
26	Joystick Y	Entrada Analógica
27	Joystick X	Entrada Analógica

Fonte: autor.

Figura 11: Sistema completo utilizado para simulação e teste



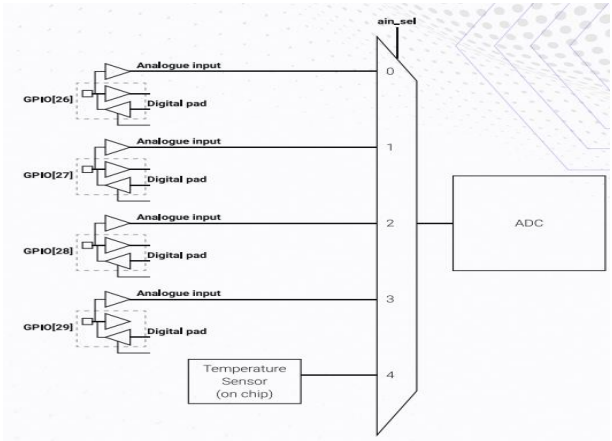
Fonte: simulador Wokwi.

Periféricos integrados ao modulo

Conversor Analógico-Digital (ADC): Os microcontroladores como o RP2040 lidam apenas com sinais digitais, porém vários sensores trabalham com sinais analógicos, para tanto é necessário converter o sinal analógico para digital, transformando o sinal continuo em um sinal discreto.

O RP2040 possui um conversor de ADC de 4 entradas, GP26 a GP29, com clock de 48MHz resolução 12-bits e 500ksps (amostras por segundo).

Figura 12: ADC do RP2040

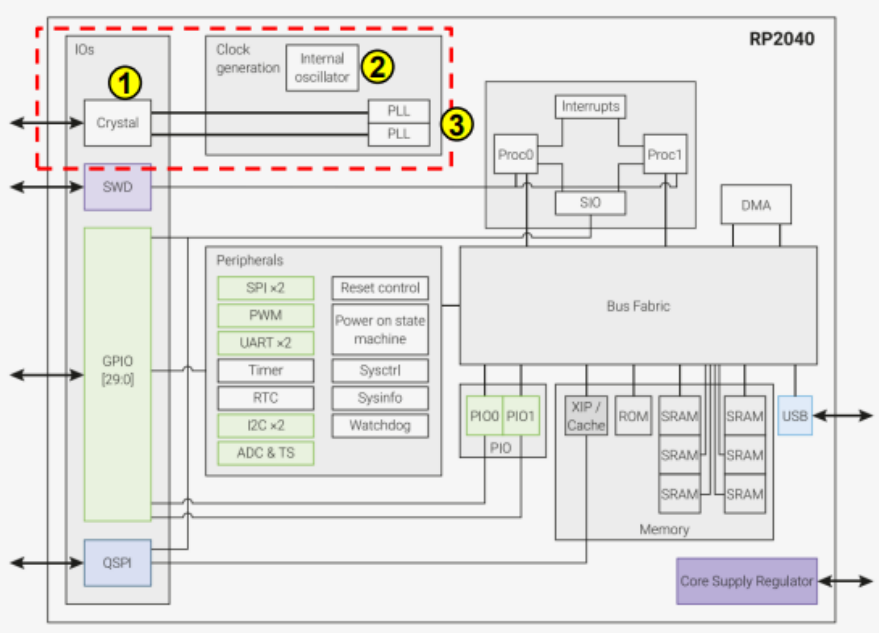


Fonte: embarcatech, 2025.

Esse conversor torna possível fazer a leitura joystick.

Phase-Locked Loops: o RP2040 possui vários subsistemas de clock que possibilitam o controle individual de frequências. Frequências podem ser ajustadas entre 12MHs e 125MHz usando as bibliotecas especificas do SDK.

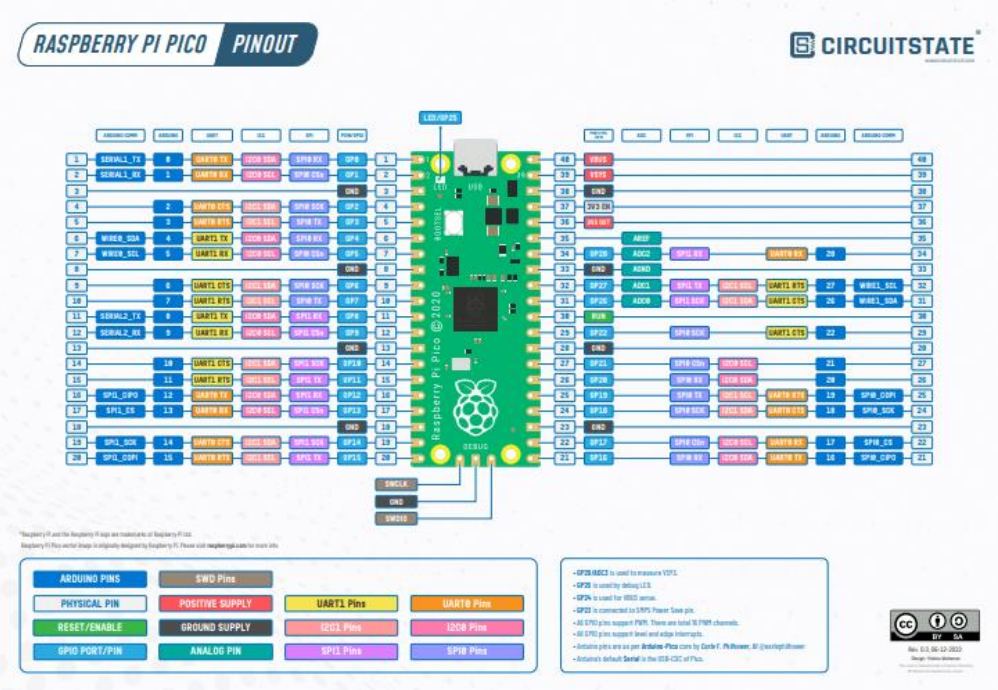
Figura 13: Arquitetura de clock do RP2040



Fonte: embarcatech, 2025.

Portas GPIO (General Purpose Input/Output): O RP2024 possui 30 GPIOs usadas para diversos propósitos, podendo ser ajustada e habilitada via software para um proposito especifico.

Figura 14: Arquitetura de entrada/saída do RP2040 ligado ao raspberry pico w

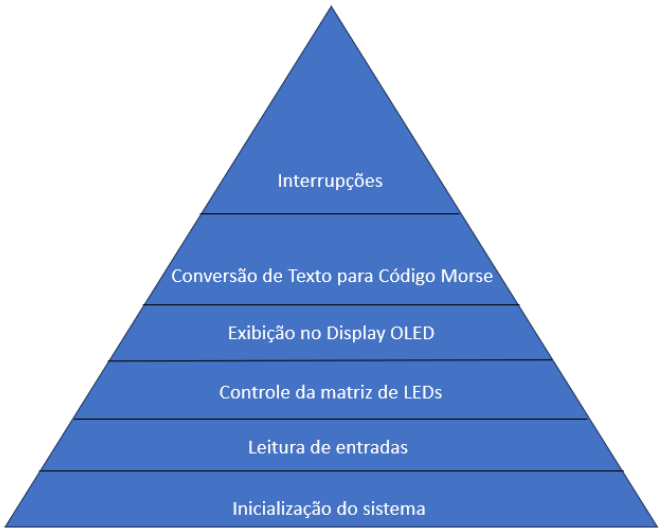


Fonte: embarcatech, 2025.

Especificação de firmware

O firmware pode ser organizado em camadas da seguinte forma:

Figura 15: Camadas do software



Fonte: autor, 2025.

Onde:

- Inicialização do sistema

A função main() inicia os periféricos e dispositivos.

npInit inicializa a PIO.

ssd1306_init inicializa o display.

ssd1306_config configura o display.

Ativa as interrupções;

Configura os pinos.

- Leitura de entradas

A função escrever() lê os valores analógicos dos eixos X e Y para determinar o movimento do cursor de acordo com a matriz caracter1 e a matriz caracter2.

Botão A apaga o último caractere digitado.

Botão B envia a string para conversão em código Morse.

Botão do joystick seleciona um caractere.

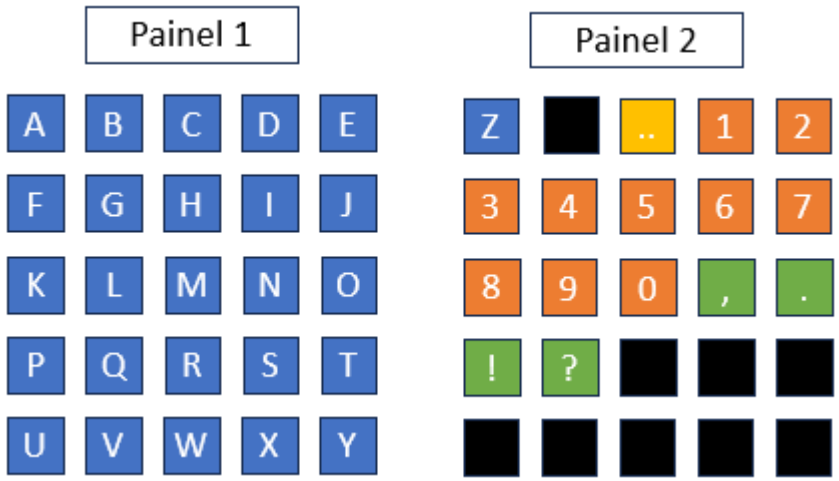
- Controle da Matriz de LEDs

A matriz de LEDs WS2818B é controlada via PIO.

A função matrizled() ajusta a matriz de acordo com o painel definido por painel1 e painel2.

Dois painéis são usados: um para letras e outro para símbolos e números.

Figura 16: Painéis com os símbolos disponíveis para a entrada



Fonte: autor, 2025.

- Exibição no Display

O display SSD1306 exibe a string digitada pelo usuário e a interface gráfica.

As funções ssd1306_draw_string() e ssd1306_pixel() desenham os caracteres e elementos gráficos.

- Conversão de Texto para Código Morse

A função tradutor (char text[80]) converte o texto digitado para código Morse, controlando o tempo através da função delay().

Os LEDs indicadores piscam conforme os sinais . e -.

gpio_put() é usado para acionar os LEDs.

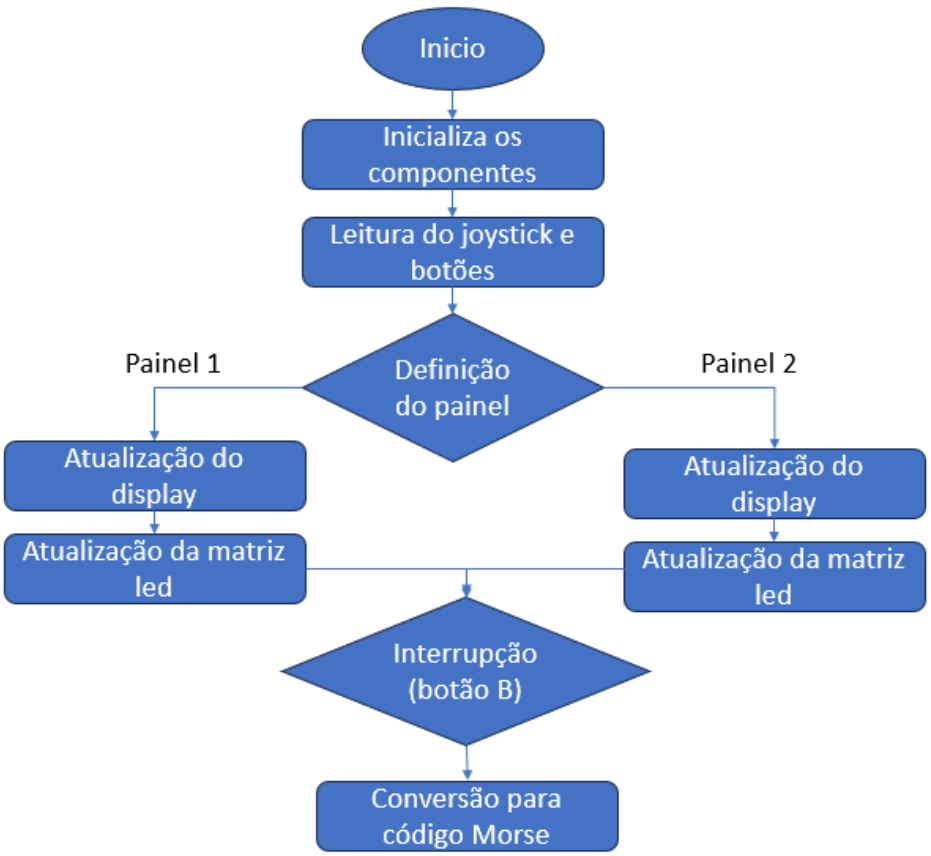
- Interrupções

A função gpio_irq_handlerA() gerencia os eventos dos botões.

Utiliza to_us_since_boot(get_absolute_time()) para implementar o debounce.

Abaixo é apresentado o fluxograma do código, de maneira simplificada ele demostra o funcionamento do sistema.

Figura 17: Fluxograma de funcionamento do firmware



Fonte: autor, 2025.

Inicialização

Inicialização é o processo que ocorre quando o dispositivo é ligado ou reiniciado. Essa fase é crucial para configurar o hardware, inicializar os periféricos e preparar o sistema para executar o código principal da aplicação. Configura os registradores do microcontrolador, definindo o modo de operação dos pinos, a velocidade do clock e outras configurações de hardware.
As Inicializações são feitas como se segue.

1. stdio_init_all(); – Inicializa a comunicação padrão.
2. i2c_init(); – Inicializa o barramento I2C.
3. adc_init(); – Inicializa os ADCs para o joystick.
4. ssd1306_init(); – Inicializa o display OLED.
5. gpio_set_irq_enabled_with_callback(); – Configura interrupções nos botões.
6. npInit(); – Inicializa a matriz de LEDs via PIO.

Registros

Registros são pequenas áreas de armazenamento dentro de um microcontrolador. Pode se dizer que existem dois tipos de registro. Registros de Propósito Geral, usados para armazenar dados temporários durante a execução de um programa e Registros de Controle, controlam o funcionamento de periféricos e outras partes do microcontrolador. Por exemplo, um registro de controle pode habilitar ou desabilitar um pino GPIO.
Abaixo é apresentada a informação referente aos registros em forma de tabela.

Tabela 2: configuração dos registros

Componente	Configuração
Display OLED	I2C @ 400kHz, Endereço 0x3C
Joystick	ADC nos GPIO26 e GPIO27
Matriz de LEDs	Controlada via PIO no GPIO7
LEDs indicadores	PWM nos GPIO11, GPIO12, GPIO13
Botões	GPIO com Pull-up Interno

Fonte: autor.

Estrutura e formato dos dados

O armazenamento dos dados se dá através de duas strings principais. O texto digitado é armazenado em char texto[80], o código Morse é armazenado em char traducao[500], onde:
. representa um ponto;
- representa um traço;
+ separa caracteres;
(espaço) separa palavras.

Protocolo de comunicação

A comunicação via UART (Universal Asynchronous Receiver/Transmitter) transmite os bits um após o outro, sequencialmente, em um único fio. Isso reduz o número de fios necessários, tornando-a ideal para comunicação de longa distância. Havendo os fios Tx, Rx e neutro entre dois dispositivos.
O UART é utilizado no projeto para enviar o código Morse convertido e o texto digitado para o terminal serial.
O I2C (Inter-Integrated Circuit) é um protocolo de comunicação serial usado para conectar dispositivos de baixa velocidade com apenas dois fios: SDA (dados) e SCL (clock). No projeto, o I2C é utilizado para enviar comandos e dados gráficos para o display OLED SSD1306.

O pacote de dados é enviado na seguinte forma para a comunicação serial via UART:
[ID] [TEXTO_DIGITADO] [MORSE_CONVERTIDO]\n

Por exemplo:
01 HELLO .... . .-.. .-.. ---

Execução do projeto

A ideia inicial do projeto surgiu junto a uma das primeiras tarefas onde foi pedido que acionássemos um led de modo que ele envia um sinal de SOS em código Morse. Para o projeto final, foi pensada na construção de um jogo da velha que pudesse funcionar na placa BitDogLab, ou a aplicação do conversor de código Morse, o conversor foi escolhido por ter uma aplicação mais científica e, apesar de “menos útil “. Foram feitas algumas pesquisas a respeito do mesmo, história, caracteres, utilidade, etc. Não foi possível encontrar muitos trabalhos semelhantes, porem em vários fóruns foi possível encontrar informações sobre. Ao final, toda a lógica do firmware de conversão foi autoral, com blocos do código sendo aproveitados de tarefas e exemplos passados.

Os testes foram feitos usando a extensão do wokwi para o IDE VScode. Todos os testes de funcionalidades foram simples, conferindo a inserção de cada caractere, imprimindo as variáveis a cada iteração, o acionamento dos botões e ajuste manual do display. Após os testes no simulador, foi usada a placa BitDogLab, notou-se duas incongruências sem explicação até o atual momento. A primeira é que o eixo x para a placa é invertido em relação ao simulador, ou seja, ao mover o joystick para a esquerda ele registra o movimento para a direita, isso foi ajustado no código para que funcionasse corretamente na placa. O outro ponto é que em momentos aleatórios ao acionar o envio da mensagem a mesma é repetida 2 vezes seguidas, o que pode se dar devido a algum buffer armazenado na memória. Optou-se por manter as funções para controle e conversão dos periféricos dentro do arquivo .c principal, pois para os ajustes e teste que foram sendo feitos durante o desenvolvimento isso tornou o processo mais prático e conveniente, além de tornar mais fácil configurar o código de acordo com a necessidade.

Perspectivas do projeto

A execução deste projeto permitiu aplicar quase todos os conhecimentos adquiridos durante o curso de sistemas embarcados, foi uma tarefa enriquecedora, à medida que problemas iam surgindo a pesquisa e improviso permitiam superá-los desenvolvendo lógicas diferentes e aplicando-as, penso que com mais tempo, seria possível aplicar mais funcionalidades e fazer mais testes para identificar possíveis bugs e falhas nas funções.

A confiabilidade do projeto parece sólida, pois atentou-se a aplicação das boas práticas de desenvolvimento de sistemas embarcados, incluindo a modularidade, uso de interrupções, debounce nos botões e comentários para o melhor entendimento do código.

Referências

Zebendo, Lúcio; Nobre, Alyn. **Código Morse.** Programa de Educação Tutorial, Universidade Federal Fluminense, Niterói, 2019.
Disponível em: https://www.telecom.uff.br/pet/petws/downloads/tutoriais/codigo_morse/Tutorial_Codigo_Morse_2019_12_19.pdf

Ventura, Layse. **Tradutor de código Morse: entenda como funciona e confira opções de conversor.** Olhar digital, 02/06/2022. Disponível em: https://olhardigital.com.br/2022/06/02/dicas-e-tutoriais/tradutor-de-codigo-morse/?utm_source=chatgpt.com

Souza, Calixto; Filho, Jesse. **Entre a escuridão e o silêncio: a relação entre as TICs e a surdocegueira utilizando a ferramenta do código morse.** Revista online de Política e Gestão Educacional, v. 21, n.1, p. 881-895, 2017 Disponível em: <http://dx.doi.org/10.22633/rpge.v21.n.esp1.out.2017.10458>. E-ISSN:1519-9029

Caetano, Thiago; Filho, Newton; Moreira, Camila. **Construção de um transmissor e de um receptor de código Morse através de sinais luminosos com uma placa Arduino**. Revista Brasileira de Ensino de Ciência e Tecnologia. v. 13, n. 2, 2020. Disponível em: https://periodicos.utfpr.edu.br/rbect/article/view/9904

Fazer o codigo morse usando string sem usar size of. Viva o Linux, 2018. Disponível em: https://www.vivaolinux.com.br/topico/C-C++/Fazer-o-codigo-morse-usando-string-sem-usar-size-of

Morse Code Implementation. GeeksforGeeks, 2023. Disponivel em: https://www.geeksforgeeks.org/morse-code-implementation/