Documentação Projeto Final Embarca Tech

ESCOPO DO PROJETO:

Título do Projeto: CipherControl - Sistema de Controle Codificado

Apresentação:

CipherControl é um sistema de controle com suporte a codificação configurável, permitindo que o usuário envie comandos codificados para controlar diferentes periféricos. O sistema também pode ser utilizado para a transferência de dados codificados, que são traduzidos e exibidos no display OLED.

Vale ressaltar que, a título de conhecimento, o sistema foi desenvolvido utilizando Código Morse como método de codificação, facilitando a adesão dos usuários. No entanto, é possível que o desenvolvedor configure previamente a correspondência entre código e letra, desenvolvendo um método de codificação personalizada para o sistema do cliente durante o desenvolvimento do firmware, caso desejado.

Objetivos:

- **Permitir codificação configurável,** onde o desenvolvedor pode alterar facilmente a correspondência entre código e letra para personalização da codificação.

- **Desenvolver um sistema de controle baseado em comandos codificados**, onde o usuário pode acionar periféricos utilizando uma codificação personalizada.

- **Oferecer um meio alternativo de transmissão de informações** que pode ser aplicado em situações de baixa largura de banda ou alta latência, como comunicações remotas e ambientes com restrições tecnológicas com baixo custo de implementação.

- **Proporcionar outra possibilidade de interação com dispositivos embarcados**, útil em cenários onde teclados ou interfaces gráficas não são viáveis.

- **Permitir a exibição de mensagens decodificadas no display OLED**, possibilitando a recepção e interpretação dos dados de forma visual.

Descrição do funcionamento:

Input via botão A: O botão A é utilizado para receber as mensagens através de código morse, onde o programa contabiliza o tempo que o botão fica pressionado para gerar os códigos “.” ou “-” e formar as letras, assim como os espaços com base na quantidade de tempo desde a última ativação do botão.

Exibição no Display OLED: O display OLED é responsável por exibir a mensagem pós decodificação, contendo tudo que foi enviado

Exibição do último caractere na matriz de LED: A cada envio de um novo caractere, o caractere em questão é representado visualmente na matriz de LED

Backspace via Botão B: O botão B age como backspace no sistema, caso o usuário perceba que enviou um caractere incorreto ele pode pressionar o botão B e o caractere será removido da mensagem no display assim como o caractere exibido na matriz de LED será alterado para o mais recente.

Controle RGB via comando codificado: O usuário pode enviar comandos específicos através das mensagens codificadas para acionar ou desativar diferentes cores do LED RGB, sendo eles: GREEN ON, GREEN OFF, BLUE ON, BLUE OFF, RED ON e RED OFF.

Justificativa:

O CipherControl se justifica pela necessidade de um meio alternativo para controle de periféricos e transmissão de dados codificados, principalmente em ambientes com restrições tecnológicas, onde são inviáveis os meios de comunicação tradicionais.

Além do controle e envio de informações por apenas um botão, CipherControl oferece uma solução barata para transmissão de dados e comandos em ambientes de alta latência. Nesses ambientes, o sistema poderia ser implementado com uma interface simples de recepção e envio de comandos através de um cabo de fibra óptica, oferecendo uma solução simples e de baixo custo, sem depender de redes complexas e protocolos pesados como TCP/IP ou LoRa.

Como permite codificação configurável, CipherControl também dificulta interceptação, além de ser uma solução educacional e experimental.

Originalidade:

O CipherControl se difere em diversos pontos dos outros projetos voltados a código morse encontrados utilizando tanto Raspberry Pi Pico quanto Arduino. Isso se dá devido ao fato desses projetos focarem exclusivamente na tradução do código morse ou para código morse, enquanto, no CipherControl, esse é apenas uma parte do sistema.

Display OLED SSD1306: O sistema oferece representações visuais, via Display OLED, das mensagens enviadas pelo usuário para facilitar a compreensão. A maioria dos projetos encontrados não ofereciam essa funcionalidade. Os poucos que ofereciam utilizavam display LCD, enquanto o CipherControl utiliza um display OLED, consumindo menos energia e permitindo uma personalização maior da forma como as mensagens são exibidas, além de um tempo de resposta menor.

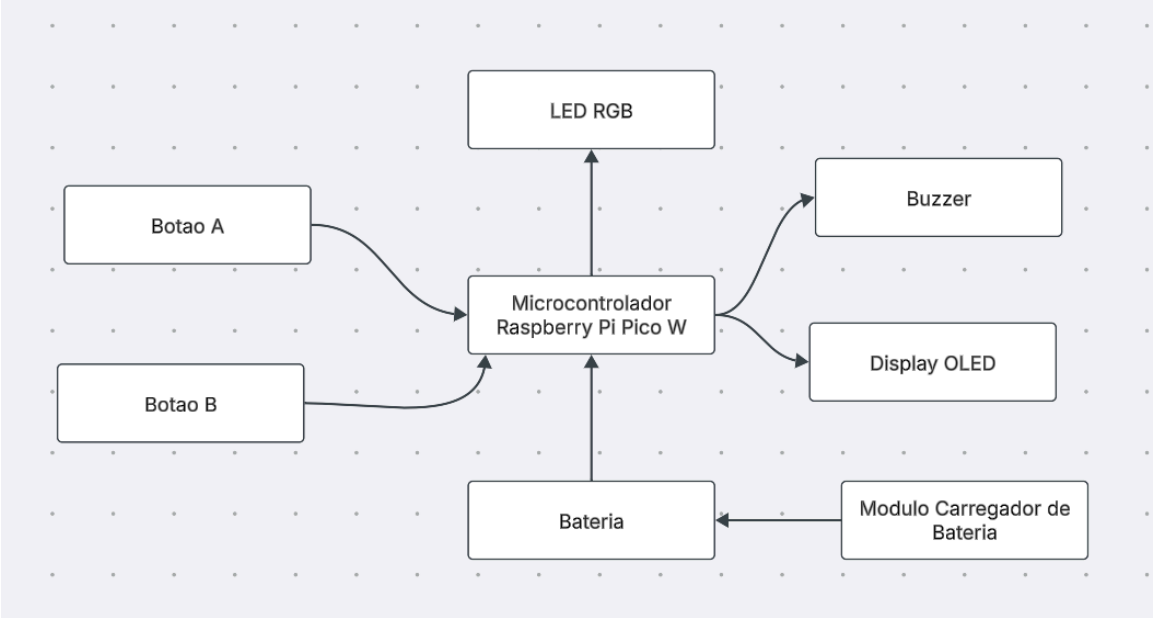
Matriz de LED WS2812: Uma forma a mais implementada para compreensão é a exibição da última letra inserida em uma matriz de leds, essa funcionalidade não foi encontrada em nenhum projeto correlato.

Backspace: Outra funcionalidade não detectada em nenhum projeto similar é a implementação do backspace, que permite o usuário apagar dígitos enviados caso tenha cometido algum erro.

Controle de Periféricos: Por fim, a última funcionalidade que não foi identificada em outros projetos disponiveis na web foi o controle de periférico através das mensagens enviadas, que é possível no CipherControl através de comandos específicos enviados.

ESPECIFICAÇÃO DO HARDWARE

Diagrama em Blocos:



Função de Cada Bloco:

Bateria: Armazena e fornece energia elétrica para o circuito, garantindo a alimentação contínua dos componentes eletrônicos, mesmo na ausência de uma fonte de energia externa.

Módulo Carregador de Bateria: Controla o processo de recarga da bateria, gerenciando a tensão e corrente de carga para otimizar a vida útil da bateria e evitar sobrecarga ou descarga excessiva.

Botão A: Responsável por enviar os pulsos do código morse para o microcontrolador.

Botão B: Aciona a interrupção responsável por agir como backspace no sistema, apagando a última letra do display e exibindo a letra mais recente na matriz de leds.

Display OLED: Exibe a mensagem enviada pelo usuário de forma decodificada.

Buzzer: É acionado enquanto o botão A está pressionado para representar os códigos que estão sendo enviados (curtos ou longos).

Led RGB: Aciona cores diferentes a depender do comando enviado pelo usuário (GREEN ON, GREEN OFF, BLUE ON, BLUE OFF, RED ON, RED OFF).

Matriz de LEDS: Exibe o caractere mais recente enviado pelo usuário.

Microcontrolador Raspberry Pi Pico W: Atua como a unidade central de processamento do sistema, executando instruções programadas para controlar e coordenar o funcionamento dos outros componentes.

Comandos e Registros Utilizados

Configuração dos Botões

```
gpio_init(BUTTON_A);
gpio_set_dir(BUTTON_A, GPIO_IN);
gpio_pull_up(BUTTON_A);
gpio_set_irq_enabled_with_callback(BUTTON_A, GPIO_IRQ_EDGE_FALL, true,
&gpio_irq_handler);
```

- **gpio_init(BUTTON_A):** Inicializa o pino do botão como GPIO.
- **gpio_set_dir(BUTTON_A, GPIO_IN):** Define o botão como entrada.
- **gpio_pull_up(BUTTON_A):** Ativa o pull-up interno para garantir que o botão funcione corretamente.
- **gpio_set_irq_enabled_with_callback(BUTTON_A, GPIO_IRQ_EDGE_FALL, true, &gpio_irq_handler):** Configura interrupção para detectar quando o botão é pressionado.

Configuração dos LEDs

```
gpio_init(LED_BLUE);
gpio_set_dir(LED_BLUE, GPIO_OUT);
```

- **gpio_init(LED_BLUE):** Inicializa o LED como GPIO.
- **gpio_set_dir(LED_BLUE, GPIO_OUT):** Define como saída digital.

Configuração do Display OLED (I²C)

Configuração da 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);
```

- **i2c_init(I2C_PORT, 400 * 1000):** Inicializa a comunicação I²C a 400kHz
- **gpio_set_function(I2C_SDA, GPIO_FUNC_I2C):** Define o pino SDA como função I²C.
- **gpio_set_function(I2C_SCL, GPIO_FUNC_I2C):** Define o pino SCL como função I²C.
- **gpio_pull_up(I2C_SDA); e gpio_pull_up(I2C_SCL):** Ativa pull-up nos pinos SDA e SCL.

Inicialização do Display OLED

```
ssd1306_init(&ssd, WIDTH, HEIGHT, false, endereco, I2C_PORT);
ssd1306_config(&ssd);
ssd1306_send_data(&ssd);
```

- **ssd1306_init():** Inicializa o display OLED.
- **ssd1306_config():** Configura o display para exibição de caracteres e gráficos.
- **ssd1306_send_data():** Envia os dados processados para o display.

Configuração do Buzzer

Configuração do PWM para o Buzzer

```
gpio_set_function(BUZZER, GPIO_FUNC_PWM);
uint slice_num = pwm_gpio_to_slice_num(BUZZER);
pwm_set_wrap(slice_num, 50000);
pwm_set_chan_level(slice_num, pwm_gpio_to_channel(BUZZER), 0);
pwm_set_enabled(slice_num, true);
```

- **gpio_set_function(BUZZER, GPIO_FUNC_PWM):** Configura o buzzer para usar PWM.
- **pwm_set_wrap(slice_num, 50000):** Define um período de onda PWM.
- **pwm_set_chan_level(slice_num, pwm_gpio_to_channel(BUZZER), 0):** Define o volume inicial como 0.
- **pwm_set_enabled(slice_num, true):** Habilita o PWM para controlar o som.

Configuração da Matriz de LED

```
PIO pio = pio0;
int sm = 0;
uint offset = pio_add_program(pio, &ws2812_program);

ws2812_program_init(pio, sm, offset, WS2812_PIN, 800000, IS_RGBW);

desenho_pio(numero_0, 0, pio, sm, 0.0, 0.0, 0.1);
```

- **pio_add_program(pio, &ws2812_program):** Adiciona o programa de controle da WS2812 à PIO.
- **ws2812_program_init(pio, sm, offset, WS2812_PIN, 800000, IS_RGBW):** Inicializa os LEDs com frequência de 800kHz.
- **desenho_pio(numero_0, 0, pio, sm, 0.0, 0.0, 0.1):** Configura um desenho inicial na matriz de LEDs.

Descrição da Pinagem Usada

Comunicação Serial (UART)

- **GP0 - RX** (Recebe dados do Serial Monitor)
- **GP1- TX** (Transmite dados para o Serial Monitor)

Display OLED SSD1306 (I2C)

- **GP14 - SDA** (Linha de Dados do barramento I2C)
- **GP15 - SCL** (Linha de Clock do barramento I2C)
- **GND - GND do OLED** (Referência comum para alimentação)
- **VSYS - VCC do OLED** (Alimentação 3.3V ou 5V para o display)

Matriz de LEDs Neopixel (WS2812)

- **GP7 - DIN (Entrada de Dados do primeiro LED Neopixel)** (Controla a matriz de LEDs WS2812)
- **VSYS - VDD de todos os LEDs Neopixel** (Fornece alimentação para a matriz de LEDs)
- **GND - VSS de todos os LEDs Neopixel** (Referência de terra para os LEDs)

Botões Push Button

- **GP5 - Terminal do botão A** (Entrada digital para leitura do botão A)
- **GP6 - Terminal do botão B** (Entrada digital para leitura do botão B)
- **GND - Outro terminal dos botões** (Para fechar o circuito quando pressionado)

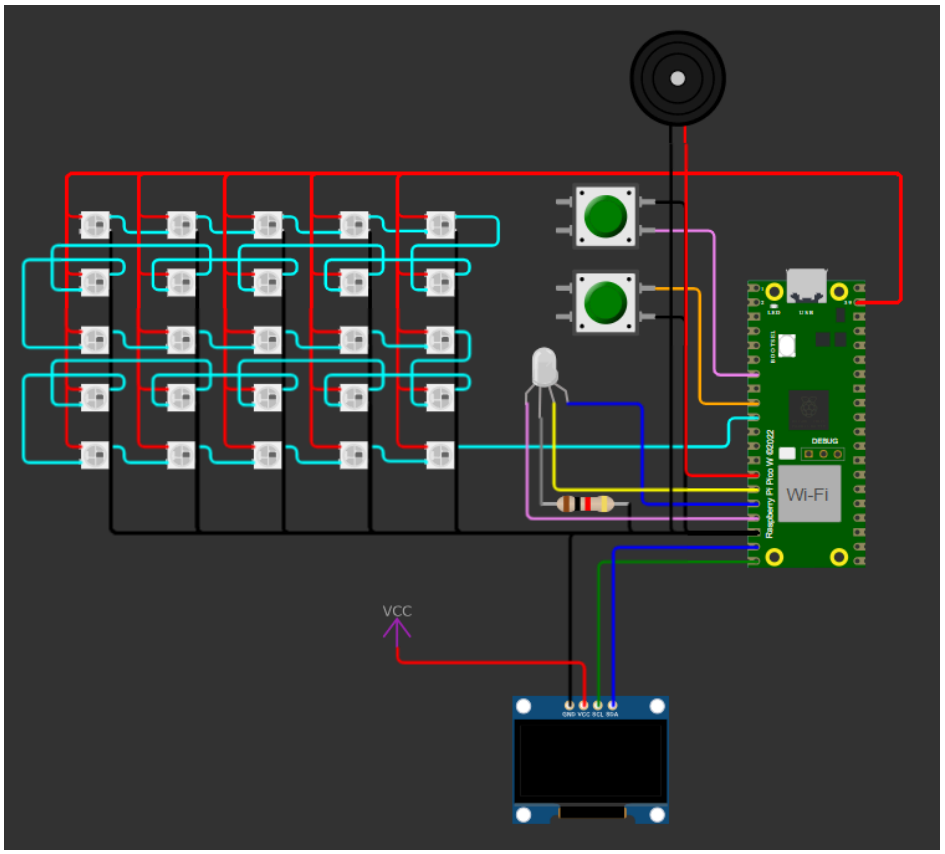
LED RGB de Cátodo Comum

- **GP13 - R (Vermelho)** (Controla a cor vermelha do LED RGB)
- **GP12 - B (Azul)** (Controla a cor azul do LED RGB)
- **GP11: -G (Verde)** (Controla a cor verde do LED RGB)
- **Resistor de 1KΩ** (Limitador de corrente conectado ao cátodo comum do LED)
- **GND - COM do LED RGB** (Pino comum do LED ligado ao terra)

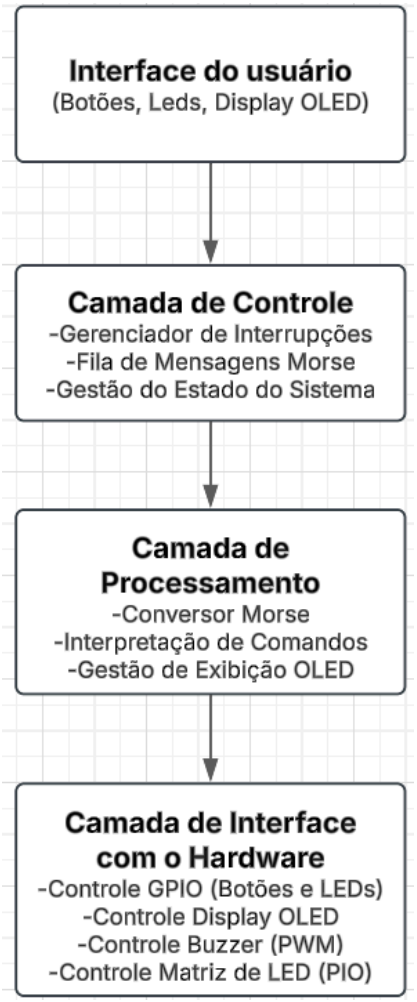
Buzzer Ativo

- **GP10 - Pino positivo do buzzer** (Envia sinais para ativar o som)
- **GND - Pino negativo do buzzer** (Fecha o circuito do buzzer para tocar o som)

Circuito Completo do Hardware:



ESPECIFICAÇÃO DO FIRMWARE



Descrição das Funcionalidades

Interface do Usuário: Permitir a interação do usuário com o sistema através de periféricos

Camada de controle: Gerenciar o fluxo de informações entre o usuário e o sistema

Camada de Processamento: Interpretar os sinais capturados e convertê-los em comandos ou mensagens legíveis.

Camada de Interface com o Hardware: Controlar os periféricos físicos do sistema

Definição das Variáveis

Variáveis de controle de Hardware

Definem pinos físicos e configurações de periféricos

```
#define I2C_PORT i2c1
#define I2C_SDA 14
#define I2C_SCL 15
#define endereco 0x3C
#define BUTTON_B 6
#define LED_BLUE 12
#define LED_RED 13
#define LED_GREEN 11
#define BUTTON_A 5
#define BUZZER 10
```

- **I2C_PORT, I2C_SDA, I2C_SCL:** Configuração dos pinos para comunicação com o display OLED.
- **BUTTON_B, BUTTON_A:** Define o pino dos botões
- **LED_BLUE, LED_RED, LED_GREEN:** Pinos dos LEDs RGB para controle via comandos Morse.
- **BUZZER**: Define o pino que controla o buzzer via PWM.

Variáveis de Controle e Estado

```
volatile bool callback_a = 0;
volatile bool callback_b = 0;
volatile char last_letter = '\0';
```

- **callback_a:** Ativado quando o botão A é pressionado (entrada Morse).
- **callback_b:** Ativado quando o botão B é pressionado (função backspace).
- **last_letter:** Guarda a última letra decodificada do código Morse.

Variáveis de Armazenamento e Processamento Morse

Buffers para entrada Morse:

```
char morse_code[10] = "";  // Código Morse de uma única letra
char message[100] = "";    // Mensagem completa
volatile int morse_index = 0;
volatile int msg_index = 0;
```

- **morse_code**: Armazena os **pontos e traços** do Morse digitado.
- **message**: Contém a **mensagem final** convertida.
- **morse_index**: Índice que rastreia a posição dentro do morse_code.
- **msg_index**: Índice que rastreia a posição dentro da message.

Controle de tempos para diferenciar pontos, traços e espaços.

```
volatile uint64_t press_time = 0;
volatile uint64_t release_time = 0;
volatile uint64_t last_press_time = 0;
volatile int first_press = 1;
volatile int new_word = 0;
```

- **press_time**: Marca o tempo quando o botão é pressionado.
- **release_time**: Marca o tempo quando o botão é solto.
- **last_press_time**: Armazena o tempo do último pressionamento para calcular pausas.
- **first_press**: Indica se esse é o primeiro pressionamento da sequência.
- **new_word**: Indica se um espaço foi detectado (início de uma nova palavra).

Inicialização

A inicialização do CipherControl começa com o carregamento das bibliotecas e definição das variáveis globais, seguido da configuração de periféricos e interrupções. Em seguida, são iniciados os buffers de armazenamento em conjunto com a tabela de conversão de morse para texto, utilizada para a decodificação dos sinais de entrada. Por fim, o software entra no loop contínuo para esperar eventos do botão.

Fluxograma

Configurações dos registros

Display OLED: É configurado para exibir a mensagem enviada pelo usuário após a decodificação e é atualizado sempre que mais sinais são enviados.

Matriz de LED: É configurado para exibir o último caractere presente na mensagem decodificada.

Botão A: É configurado para enviar os sinais que serão interpretados e codificados pelo programa, assim como para acionar o buzzer enquanto pressionado.

Botão B: É configurado para remover o último caractere da mensagem quando acionado e atualizar o display e a matriz com a mensagem atualizada.

Buzzer: É configurado para emitir um som enquanto o Botão A estiver pressionado

LED RGB: É configurado para ligar ou desligar cores específicas caso a mensagem corresponda a um dos comandos definidos.

Estrutura e formato dos dados

Os dados utilizados no CipherControl são capturados a partir dos botões pressionados pelo usuário que são processados como código morse, convertida em caracteres e armazenadas. Isso é feito através do tempo de pressionamento do botão, registrado nas variáveis press_time, release_time e last_pressed_time.

Com base nesses valores é definido se deve ser registrado um traço, ponto ou espaço onde são armazenados no buffer morse_code[] e a mensagem decodificada completa é armazenada no message[].

Protocolo de Comunicação

O sistema utiliza o protocolo de comunicação I2C para transmissão de dados para o display OLED, permitindo o envio de dados gráficos para a representação visual da mensagem recebida.

EXECUÇÃO DO PROJETO

Metodologia

1. **Definição do hardware:** O hardware foi definido por padrão da instituição para a elaboração do projeto

2. **Pesquisa e brainstorming**: Foram realizadas diversas pesquisas sobre diferentes tópicos e projetos para a definição do tema do projeto com base no hardware disponivel

3. **Definição dos escopos e funcionalidades:** Após a definição do tema foram definidas todas as funcionalidades e aplicações do sistema

4. **Desenvolvimento:** Após a definição clara do projeto foi iniciada a fase de desenvolvimento do projeto

5. **Depuração:** Ao longo do desenvolvimento houveram diversas necessidades de depuração do código

Teste De Validação

1. **Envio recepção e tradução do código morse:** Todos os caracteres foram testados com o envio individual de todos os caracteres para garantir o funcionamento correto da tradução e recepção de informações

2. **Display e Matriz:** Ambas as ferramentas para representação visual da mensagens foram testadas também com o envio de todos os caracteres, além de testar o funcionamento do backspace garantindo a alteração correta dos estados de ambos.

3. **LED RGB:** No LED RGB, foram enviados todos os diferentes comandos, checando se a reação correspondia ao comando enviado de cada um.

Resultados

Os testes de validação confirmam a precisão e confiabilidade do CipherControl. A tradução do código Morse foi validada com o envio individual de todos os caracteres, garantindo uma interpretação correta.

A exibição no display OLED e na matriz de LEDs funcionou sem erros, incluindo a remoção de caracteres via backspace. O controle do LED RGB respondeu corretamente a todos os comandos enviados, assegurando a execução precisa das ações. Os resultados indicam que o sistema opera de forma estável e eficiente, garantindo tradução, exibição e controle precisos.

Referências

[EOG single switch morse code translate input device for individuals with the motor neuron disease]
[Morse Code Detector and Decoder using Eye Blinks]