Título: Comunicação Alternativa - TalkGo

Aluno: Wasley dos Santos Silva

Projeto no Wokwi: https://wokwi.com/projects/422002755613637633

Vídeo de teste na BitDogLab: https://www.youtube.com/watch?v=KjrngxYopJA

1. ESCOPO DO PROJETO

1.1 Apresentação do Projeto

A comunicação é um aspecto fundamental da vida, permitindo a interação e expressão de pensamentos e emoções. No entanto, muitas pessoas com deficiências motoras ou na fala enfrentam dificuldades significativas para se comunicar, o que pode impactar sua qualidade de vida. O projeto TalkGo surge como uma solução acessível e inclusiva, utilizando a placa BitDogLab e frases pré-configuradas conforme a necessidade de cada pessoa para possibilitar a comunicação de forma eficiente e prática.

Público-Alvo: Idosos com Dificuldades Motoras, Pessoas com Paralisia Cerebral, Esclerose Múltipla, Acidente Vascular Cerebral (AVC), Doenças Neuromusculares, Esclerose Lateral Amiotrófica (ELA).

1.2 Objetivos

- Dar voz às pessoas que enfrentam dificuldades na comunicação, seja por limitações na fala ou na locomoção. Um sistema acessível para auxiliar a comunicação dessas pessoas.
- Permitir que frases sejam configuráveis de acordo com as necessidades individuais.
- Utilizar a placa BitDogLab para oferecer uma solução prática e de baixo custo usando a maioria de seus periféricos.
- Inicialmente, utilizaremos um joystick, com planos para, no futuro, implementar o reconhecimento de sons por meio de um microfone integrado e a possibilidade de gerar voz ao selecionar a frase desejada, também pensei nos pacientes com Esclerose Lateral Amiotrófica com a implementação de um sensor

Sensor Capacitivo Touch TTP223B, isso porque esses pacientes ao longo do tempo perdem grande das suas capacidades motoras.

1.3 Descrição do Funcionamento

O TalkGo é um sistema baseado no Raspberry Pi Pico W que auxilia na comunicação de pessoas com dificuldades motoras e/ou na fala. Ele permite ao usuário selecionar mensagens pré-definidas ou formar palavras através de um joystick e botões, exibindo o texto selecionado em um display LCD.
Fluxo de Operação:

1 **Início:** O dispositivo é ligado e exibe a tela inicial com o nome do sistema e a temperatura ambiente.
2 **Navegação pelo Menu:** O usuário movimenta o joystick para cima ou para baixo para navegar entre as categorias de mensagens.
3 **Seleção de Opção:** Ao pressionar o botão do joystick, o usuário entra no submenu e pode escolher uma mensagem específica.
4 **Exibição da Mensagem:** A frase escolhida é exibida no LCD OLED (SSD1306), possibilitando a comunicação com terceiros.
5 **Modo Alfabeto:** O usuário pode construir palavras letra por letra, confirmando cada uma com o botão A.
6 **Sinal de Emergência:** Ao pressionar o botão B, a mensagem "SOCORRO" aparece no display, e um alerta sonoro e visual é ativado.

1.4 Justificativa

Muitas pessoas com limitações motoras ou na fala enfrentam dificuldades para se comunicar, o que pode resultar em isolamento social e dificuldades no dia a dia e até um menor tempo de vida. Os sistemas existentes no mercado geralmente são caros e de difícil acesso, dificultando ainda mais a inclusão dessas pessoas. O TalkGo se propõe a ser uma alternativa acessível e

personalizável, permitindo que o usuário se expresse de forma prática e intuitiva por meio de um joystick, Leds indicadores, botão de emergência e um display OLED.

1.5 Originalidade

A principal inspiração para o desenvolvimento deste projeto foi o filme “A Teoria de Tudo”, que retrata a vida de Stephen Hawking diagnosticado com Esclerose Lateral Amiotrófica (ELA). No filme, o protagonista enfrenta a perda de todos os movimentos e da fala, enfrentando enormes dificuldades para se comunicar ao longo de sua vida, mesmo sendo um gênio. Diante dessa perspectiva, a motivação central se tornou como ajudar pessoas que enfrentam desafios semelhantes.

Com isso, realizei uma pesquisa sobre projetos relacionados ao tema e explorei maneiras de aplicar esses conhecimentos utilizando minha principal ferramenta: a placa BitDogLab.

O artigo “Comunicação para ELA utilizando Arduino” apresenta um projeto inovador destinado a ajudar pessoas com Esclerose Lateral Amiotrófica (ELA) a se comunicarem. A proposta envolve o uso de um sistema que combina tecnologia de rastreamento ocular e um microcontrolador Arduino, permitindo que movimentos oculares sejam traduzidos em comandos, funcionando como um teclado virtual. Essa solução proporciona maior autonomia e qualidade de vida aos usuários, capacitando-os a expressar suas necessidades e interagir socialmente. Além disso, o projeto é descrito como uma alternativa acessível e de baixo custo, fomentando a personalização de dispositivos de comunicação.

Outro artigo é do Diário do Nordeste que destaca um programa inovador desenvolvido em Fortaleza, que visa facilitar a comunicação de pessoas com dificuldades na fala, como aquelas que enfrentam condições como a Esclerose Lateral Amiotrófica (ELA). O programa utiliza tecnologia assistiva para permitir que os usuários se comuniquem de maneira mais eficaz, por meio de dispositivos e aplicativos que interpretam gestos e movimentos oculares. Além de ampliar a autonomia das pessoas com limitações na fala, a iniciativa busca promover a

inclusão social, proporcionando melhores oportunidades de interação e expressão.

2. ESPECIFICAÇÃO DO HARDWARE

2.1 Diagrama de blocos

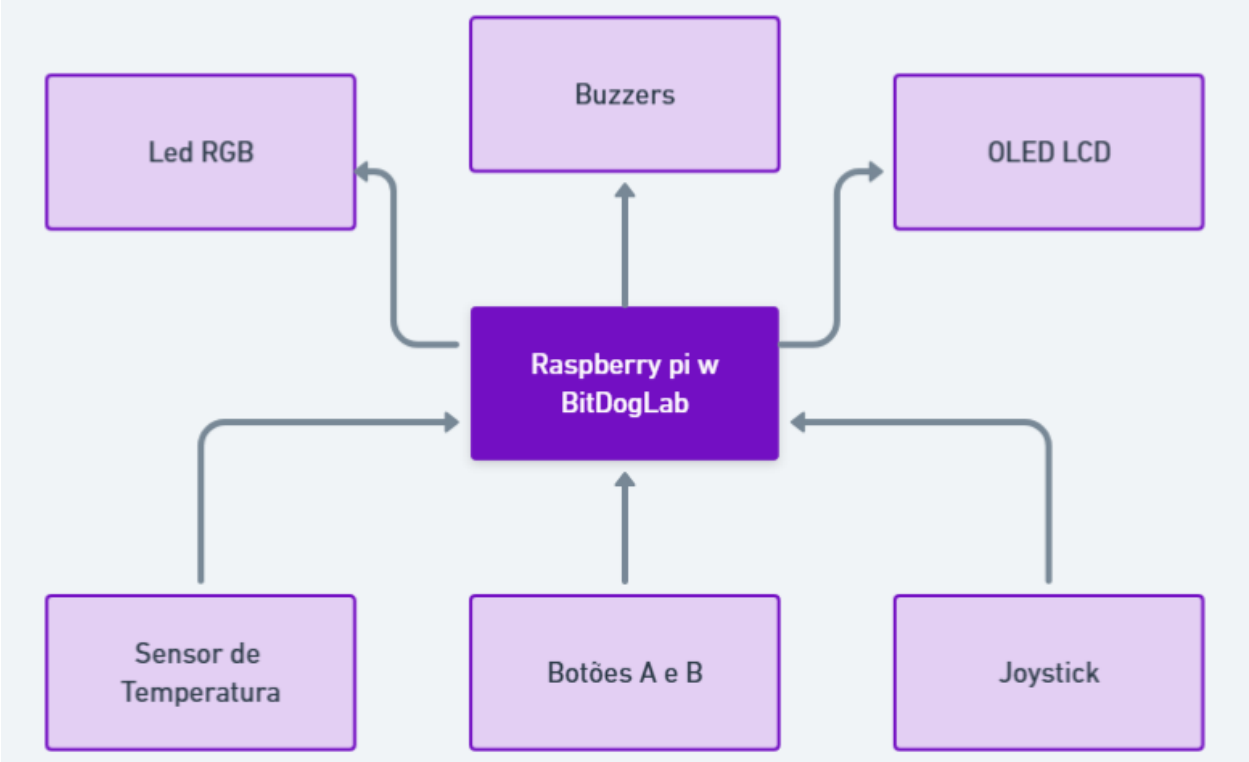


Figura 1 - Diagrama do projeto

2.2 Função de cada bloco

Inicialização: Configura todos os pinos de entrada e saída, inicializa o ADC, o barramento I2C e o display OLED. Este bloco é responsável por preparar o sistema para operação.

Leitura de Sensores: Lê os valores do joystick (eixos X e Y) e a temperatura do sensor. Converte os valores lidos do ADC para uma representação utilizável (por exemplo, temperatura em graus Celsius).

Menu e Navegação: Gerencia a exibição do menu principal e dos submenus. Permite ao usuário navegar entre as opções usando o joystick e selecionar uma opção pressionando o botão central do joystick.

Exibição de Dados: Renderiza texto e informações no display OLED. Exibe mensagens, a temperatura atual e feedback visual sobre as ações do usuário. **Controle de LEDs e Buzzer**: Controla os LEDs e o buzzer para fornecer feedback visual e sonoro ao usuário. Os LEDs indicam o estado atual do menu ou a seleção, enquanto o buzzer emite sons para confirmações ou alertas.

2.3 Descrição da pinagem usada

Componente	Pino do Raspberry Pi Pico	Função
Joystic VRX	GP26 (ADC0)	Leitura do eixo X
Joystic VRY	GP27 (ADC1)	Leitura do eixo Y
Joystic SW	GP22	Botão de seleção
Display OLED SSD1306 SDA	GP14	Comunicação I2C
Display OLED SSD1306 SCL	GP15	Comunicação I2C
LED Vermelho	GP13	Indicação visual
LED Verde	GP11	Indicação visual
LED Azul	GP12	Indicação visual
Buzzer	GP10	Alerta sonoro
Botão A	GP5	Emergência
Botão B	GP6	Finalizar Frase

2.4 Configuração de cada bloco

Definição de Menus e Opções

```
// Definição das opções de menu
const char *menus[][6] = {
    {"Voltar"},
    {"Ajuda", "Muita dor", "Enjoado", "Desorientado", "Mudar posicao"},
    {"Agua", "Comer", "Ao Banheiro", "Tomar banho", "Passear"},
    {"Feliz", "Triste", "Com medo", "Cansado", "Dormir"},
    {"Incrivel", "Obrigado", "Adoro voce", "Otimo trabalho", "Me faz feliz"}
};

// Definição dos títulos do menu principal
const char *menu_titles[] = {" TalkGo ", "Emergencia", "Necessidades", "Emocoes", "Elogios"};
```

Desenhar Texto no Display OLED

```
// Função para desenhar uma string no display SSD1306 com escala
void ssd1306_draw_string_scaled(uint8_t *buffer, int x, int y, const char *text, int scale) {
    while (*text) {
        for (int dx = 0; dx < scale; dx++) {
            for (int dy = 0; dy < scale; dy++) {
                ssd1306_draw_char(buffer, x + dx, y + dy, *text);
            }
        }
        x += 6 * scale;  // Avança a posição do caractere
        text++;
    }
}
```

Exibir Menu no console

```
/ Função para exibir o menu no console
void print_menu() {
    if (in_submenu) {
        printf("\n%s:\n", menu_titles[current_menu]);
        for (int i = 0; i < 6; i++) {
            if (menus[current_menu][i][0] != '\0') {
```

```
                if (i == menu_index) {
                    printf("-> %s\n", menus[current_menu][i]);
                } else {
                    printf("   %s\n", menus[current_menu][i]);
                }
            }
        }
    } else {
        printf("**************************\n");
        for (int i = 0; i < 5; i++) {
            if (i == current_menu) {
                printf(">> %s\n", menu_titles[i]);
            } else {
                printf("   %s\n", menu_titles[i]);
            }
        }
        printf("**************************\n");
    }
}
```

Gerar Tom no Buzzer

```
// Função para gerar um tom no buzzer
void tone(unsigned int frequency, unsigned int duration) {
    unsigned long period = 1000000 / frequency; // Período em
microssegundos
    unsigned long end_time = time_us_64() + (duration * 1000); //
Tempo final

    while (time_us_64() < end_time) {
        gpio_put(BUZZER_PIN, 1); // Liga o buzzer
        sleep_us(period / 2);
        gpio_put(BUZZER_PIN, 0); // Desliga o buzzer
        sleep_us(period / 2);
    }
}
```

Controle de LEDs

```
// Função para atualizar os LEDs com base no menu selecionado
void update_leds() {
    if ((current_menu == 1 && menu_index <= 4) ||
        (current_menu == 2 && menu_index <= 3)) {
```

```
            gpio_put(LED_RED, true);   // Ativa LED vermelho
            gpio_put(LED_GREEN, false); // Desativa LED verde
            gpio_put(LED_BLUE, false);  // Desativa LED azul
        } else if ((current_menu == 3 && menu_index <= 3) ||
                   (current_menu == 4 && menu_index <= 3)) {
            gpio_put(LED_GREEN, true);  // Ativa LED verde
            gpio_put(LED_RED, false);    // Desativa LED vermelho
            gpio_put(LED_BLUE, false);   // Desativa LED azul
        } else {
            gpio_put(LED_RED, false);    // Desativa LED vermelho
            gpio_put(LED_GREEN, false);   // Desativa LED verde
            gpio_put(LED_BLUE, true);     // Ativa LED azul
        }
    }
```

Conversão do Sensor de Temperatura

```
    // Função para converter o valor lido do ADC para temperatura em
    graus Celsius
    float ler_adc_para_temperatura(uint16_t valor_adc) {
        const float fator_conversao = 3.3f / (1 << 12);  // Conversão
    de 12 bits (0-4095) para 0-3.3V
        float tensao = valor_adc * fator_conversao;      // Converte o
    valor ADC para tensão
        float temperatura = 27.0f - (tensao - 0.706f) / 0.001721f;  //
    Converte a tensão para temperatura em Celsius
        return temperatura;
    }
```

2.5 Descrição do funcionamento

Controle do Joystick

O joystick é um sensor analógico, conectado ao ADC do Raspberry Pi Pico W: Eixo X: VRX_PIN (GPIO 26), Eixo Y: VRY_PIN (GPIO 27), Botão Central: SW_PIN (GPIO 22, entrada digital com pull-up).

Valores lidos pelo ADC são convertidos para definir a direção do movimento no menu.

Código para leitura do joystick:

```c
adc_select_input(0);  // Seleciona o canal do joystick X
        uint16_t vrx_value = adc_read();  // Lê o valor do eixo X
        adc_select_input(1);  // Seleciona o canal do joystick Y
        uint16_t vry_value = adc_read();  // Lê o valor do eixo Y
        bool sw_pressed = !gpio_get(SW_PIN);  // Verifica se o botão switch foi pressionado

```

Exibição no Display OLED (SSD1306 via I2C)

O display SSD1306 é controlado por I2C nos pinos:

- SDA: GPIO 14
- SCL: GPIO 15

Para exibir mensagens, utilizamos a biblioteca ssd1306.h, enviando caracteres formatados.

Código para exibir texto no display:

```c
memset(ssd, 0, ssd1306_buffer_length);  // Limpa o display
                    render_on_display(ssd, &frame_area);
                    ssd1306_draw_string_scaled(ssd, 20, 30, "SOCORRO", 2);  // Exibe mensagem de socorro
                    render_on_display(ssd, &frame_area);
```

Acionamento dos LEDs Indicadores

Os LEDs são usados para indicar o estado do menu:

- LED Vermelho (GPIO 13) → Emergências
- LED Verde (GPIO 11) → Sentimentos e elogios
- LED Azul (GPIO 12) → Outras opções

Código para controle dos LEDs:

```c
void update_leds() {
    if ((current_menu == 1 && menu_index <= 5) || (current_menu == 2 && menu_index <= 5)) {
        gpio_put(LED_RED, true);   // Ativa LED vermelho
        gpio_put(LED_GREEN, false); // Desativa LED verde
        gpio_put(LED_BLUE, false);  // Desativa LED azul
    } else if ((current_menu == 3 && menu_index <= 5) || (current_menu == 4 && menu_index <= 5) || (current_menu == 5 && menu_index <= 23)) {
        gpio_put(LED_GREEN, true);  // Ativa LED verde
        gpio_put(LED_RED, false);    // Desativa LED vermelho
        gpio_put(LED_BLUE, false);   // Desativa LED azul
    } else {
        gpio_put(LED_RED, false);    // Desativa LED vermelho
        gpio_put(LED_GREEN, false);   // Desativa LED verde
        gpio_put(LED_BLUE, true);     // Ativa LED azul
    }
}
```

Acionamento do Buzzer

O buzzer (GPIO 10) é ativado para emitir sons de alerta quando o botão B é pressionado.

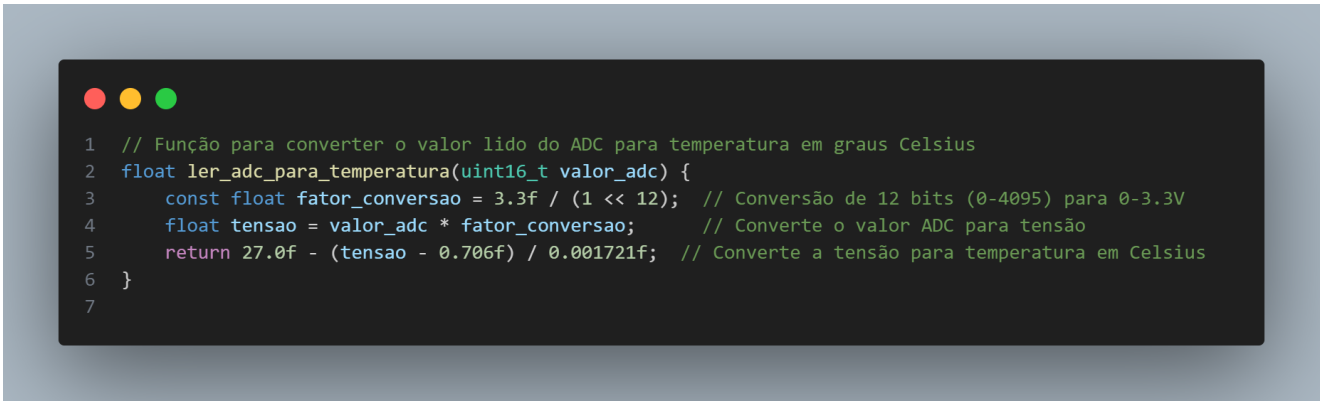
Código para ativar o alerta:

```c
for (int i = 0; i < 10; i++) {  // Toca um som de alerta
                    gpio_put(LED_GREEN, false);
                    gpio_put(LED_BLUE, false);
                    gpio_put(LED_RED, true);
                    tone(600, 200); //acionando o buzzer
```

Registro da Temperatura

A temperatura é medida pelo sensor embutido do RP2040, convertendo a leitura do ADC em graus Celsius.

Código para conversão da temperatura:



```c
// Função para converter o valor lido do ADC para temperatura em graus Celsius
float ler_adc_para_temperatura(uint16_t valor_adc) {
    const float fator_conversao = 3.3f / (1 << 12);  // Conversão de 12 bits (0-4095) para 0-3.3V
    float tensao = valor_adc * fator_conversao;      // Converte o valor ADC para tensão
    return 27.0f - (tensao - 0.706f) / 0.001721f;  // Converte a tensão para temperatura em Celsius
}

```

2.6 Circuito completo do hardware

A seguir, os periféricos usados no simulador e o circuito completo da placa do projeto:

1. 1 Placa Raspberry Pi W
2. LED RGB (Pino R (vermelho), Pino G (verde), Pino B (azul))
3. 2 Botões
4. 1 Buzzer
5. 1 Display OLED 128x64
6. 1 Joystick(Eixos X e Y, Botão central)
7. 3 resistores 220W

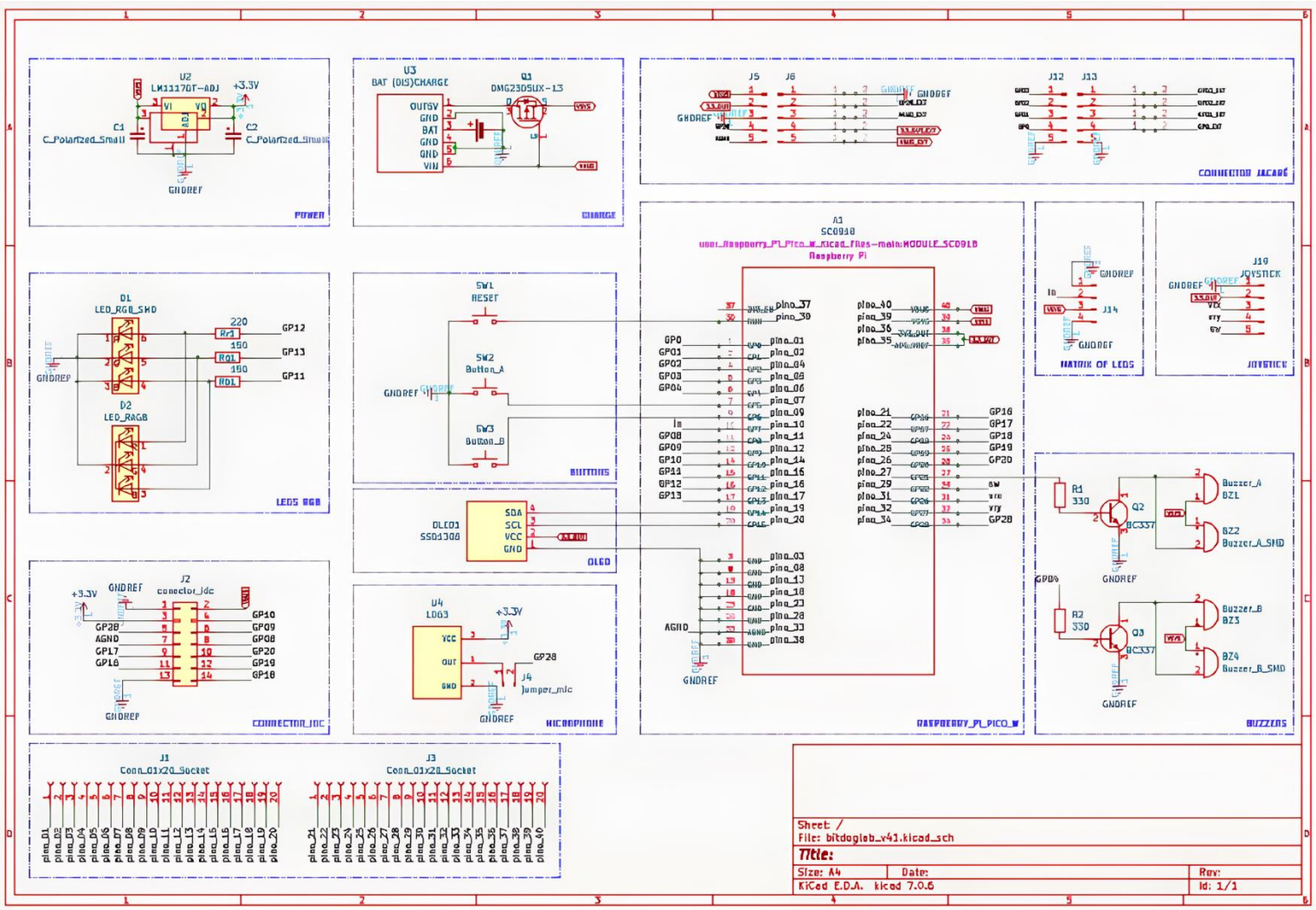


Figura 2 - Circuito completo do projeto

Os periféricos usados no simulador foram escolhidos para que o projeto fosse igual a BitDogLab, buscando utilizar a maioria dos componentes disponíveis na placa.

3. ESPECIFICAÇÃO DO FIRMWARE

3.1 Blocos Funcionais

Aplicação: Responsável pela interação com o usuário, exibindo menus e capturando seleções, leds e sinais de alerta.

| Aplicação (Interação do Usuário)
| Exibição do menu no OLED
| Leitura do joystick e botões
| Feedback sonoro e visual (LEDs e buzzer)

Camada de Controle: Gerencia a lógica do menu, interpreta as entradas do usuário e executa ações.

| Camada de Controle
| Processamento da navegação no menu
| Seleção e exibição de mensagens
| Detecção de emergência (BTN_A)

Comunicação com Periféricos: Interage diretamente com os componentes físicos, lendo sensores e acionando atuadores.

| Camada de Comunicação com Periféricos
| Interface com display OLED via I2C
| Leitura de valores do joystick (ADC)
| Controle de LEDs e buzzer (GPIO)
| Medição de temperatura (ADC)

Hardware: Inclui o Raspberry Pi Pico W e todos os dispositivos conectados, como joystick, display OLED, LEDs e buzzer.

| Hardware (Raspberry Pi Pico W e Sensores)
| Microcontrolador
| Joystick, botões
| Display OLED, buzzer, LEDs

3.2 Descrição das Funcionalidades

- **Inicialização:** Configura os pinos de entrada e saída, inicializa o ADC e o display OLED.

- **Leitura de Sensores:** Lê os valores do joystick e converte a leitura do ADC para temperatura.
- **Menu e Navegação:** Permite ao usuário navegar entre diferentes opções de comunicação usando o joystick.
- **Exibição de Dados:** Mostra mensagens e a temperatura no display OLED.
- **Controle de LEDs e Buzzer:** Acende LEDs e toca sons para feedback ao usuário.
- **Comunicação de Emergência:** Permite ao usuário enviar um sinal de socorro através de um botão.]

3.3 Definição das Variáveis

Variáveis:

```
bool in_submenu = false; //Indica se estamos no submenu
int current_menu = 0; //Índice do menu principal
int menu_index = 0; //Índice do submenu
int last_vry_value = 0; //Último valor do sensor de movimento
bool sw_pressed = false; //Estado do botão switch
bool last_sw_state = false; //Último estado do botão switch
bool menu_updated = false; //Indica se o menu foi atualizado
int vry_value; //Valor atual do sensor de movimento
bool selecting_option = false; //Para saber se estamos selecionando uma opção
```

3.4 Fluxograma

Na Figura 4, é apresentado o fluxograma do projeto. O processo se inicia com a exibição do nome no LCD. Quando o usuário começa a utilizar o joystick, ele pode navegar pelas opções do menu principal. Ao pressionar o botão central do joystick, o usuário seleciona a opção desejada. Em seguida, as opções do submenu são exibidas. Após essa seleção, a mensagem correspondente

aparece na tela LCD, acompanhada de um sinal sonoro do buzzer e da ativação de um LED que indica se a mensagem é urgente ou não.

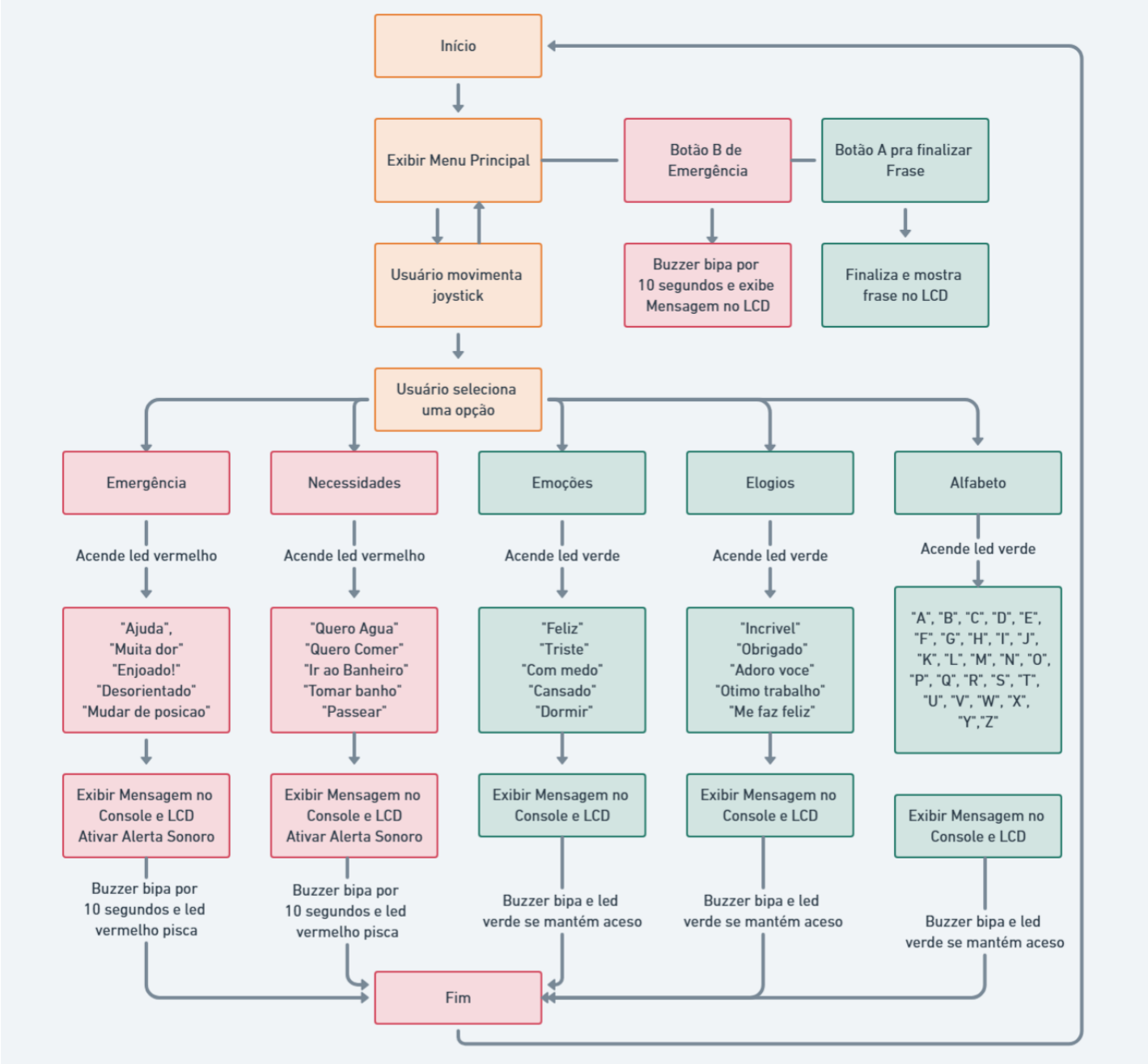


Figura 3- Fluxograma da aplicação

3.5 Inicialização

A inicialização do firmware:

- Configuração dos pinos de I/O.
- Inicialização do ADC e do display OLED.
- Configuração do barramento I2C.
- Inicialização dos LEDs e do buzzer.

3.6 Estrutura e Formato dos Dados

Os dados são organizados em estruturas e arrays:

- Menus: Arrays de strings que contêm as opções de menu.
- Estruturas: `render_area` para definir a área de renderização do display.

3.7 Organização da Memória

A memória é organizada da seguinte forma:

- Variáveis Globais: Armazenam estados e configurações do sistema.
- Buffers: Usados para armazenar dados a serem exibidos no display OLED.
- Estruturas: Usadas para organizar dados relacionados, como a área de renderização.

3.8 Protocolo de Comunicação

O firmware utiliza o protocolo I2C para comunicação com o display OLED. As funções de inicialização e renderização são responsáveis por enviar dados ao display.

3.9 Formato do Pacote de Dados

Os pacotes de dados enviados ao display OLED contêm:

- Comando: Indica a operação a ser realizada (ex: limpar display, desenhar texto).

- Dados: Contém as informações a serem exibidas, como a string de texto e suas coordenadas.

4. EXECUÇÃO DO PROJETO

4.1 Metodologia

O projeto "TalkGo" começou com uma pesquisa detalhada sobre as necessidades de comunicação de pessoas com deficiências motoras e na fala. Essa revisão incluiu artigos com propostas relacionadas, revelando que muitas soluções de comunicação alternativa existentes são caras e complexas, limitando o acesso aos que mais precisam. Condições como paralisia cerebral, esclerose múltipla e AVC foram identificadas como principais questões que afetam a comunicação dessas pessoas.

Após a pesquisa, foi escolhido o microcontrolador Raspberry Pi Pico devido à sua relação custo-benefício e facilidade de programação. O projeto também optou por um joystick analógico para navegação intuitiva e um display OLED SSD1306 para exibir mensagens. LEDs e um buzzer foram integrados ao sistema, proporcionando feedback visual e sonoro. As funcionalidades do software foram definidas para permitir navegação por menus e seleção de mensagens pré-configuradas, com categorias como "Emergência", "Necessidades" e "Emoções", tudo foi visando usar a nossa principal placa do projeto que é a BitDogLab que possui todos esses periféricos.

A IDE VScode foi utilizada para o desenvolvimento, suportando programação em C. O código foi construído em etapas, iniciando pela configuração do hardware e implementação do menu, submenu. Testes frequentes foram realizados através da placa BitDogLab o que auxiliou na identificação e correção de bugs, garantindo que todas as funcionalidades estivessem operacionais antes da conclusão do projeto. O vídeo do teste na placa foi disponibilizado no início desse relatório.

4.2 Testes de validação

Testes de Funcionalidade: Cada funcionalidade foi testada individualmente. Isso incluiu a navegação pelo menu, a seleção das opções e a exibição de mensagens no display, e os botões A e B. O feedback visual e sonoro foi verificado para garantir que o usuário recebesse as confirmações adequadas.

Testes de Usabilidade: O sistema foi testado por amigos e colegas de trabalho, diante do prazo curto não tive oportunidade de testar com pacientes com alguma dessas doenças. Ajustes foram feitos com base nas sugestões recebidas. O vídeo do teste na placa está no começo desse relatório, onde mostro todos os testes realizados.

4.3 Discussão dos Resultados

Os resultados dos testes mostraram que o projeto "TalkGo" atende às necessidades de comunicação de pessoas com deficiências motoras e na fala. O sistema é intuitivo e fácil de usar, permitindo que os usuários naveguem pelos menus e selecionem mensagens de forma eficaz. O feedback visual e sonoro proporciona uma experiência de usuário positiva.

Em conclusão, o projeto " TalkGo" não apenas atende às necessidades identificadas, mas também oferece uma solução acessível e de baixo custo, contribuindo para a melhoria da qualidade de vida das pessoas com deficiências motoras e na fala. O sucesso do projeto abre caminho para futuras melhorias e expansões, como a adição de novas funcionalidades para melhorar ainda mais e atender mais pessoas como sistema de voz para as mensagens selecionadas e implementação de IA para reconhecimento de expressões de pacientes que possuem nenhuma mobilidade, voz ou movimento.

REFERÊNCIAS

AFASIA: entenda as dificuldades de comunicação após um AVC. IstoÉ. Disponível em: https://bemestar.istoe.com.br/afasia-entenda-as-dificuldades-de-comunicacao-apos-um-avc/. Acesso em: 7 fev. 2025.

BITDOGLAB. Documentação do BitDogLab. Disponível em: https://github.com/BitDogLab/BitDogLab/tree/main/doc. Acesso em: 15 fev. 2025.

DIÁRIO DO NORDESTE. Programa busca facilitar a comunicação de pessoas com dificuldade na fala. Disponível em: https://diariodonordeste.verdesmares.com.br/metro/programa-busca-facilitar-a-comunicacao-de-pessoas-com-dificuldade-na-fala-1.3090774. Acesso em: 11 fev. 2025.

EMBARCADOS. BitDogLab: uma jornada educativa com eletrônica, embarcados e IA. Disponível em: https://embarcados.com.br/bitdoglab-uma-jornada-educativa-com-eletronica-embarcados-e-ia/. Acesso em: 15 fev. 2025.

NASCIMENTO, L. T.; GOMES, A. M. S.; CUNHA, A. R. A. A relação entre o desenvolvimento da linguagem e a função executiva em crianças. Revista CEFAC. v. 22, n. 4, p. 1-8, 2020. Disponível em: https://www.scielo.br/j/rcefac/a/xJ8CkPHy7V96sBw6XfJTLMH/?format=pdf&lang=pt. Acesso em: 8 fev. 2025.

SOUZA, Franky. A comunicação como fomentadora de inclusão: Um estudo sobre acessibilidade e participação social. 2017. Disponível em: https://laticsufcg.blogspot.com/2017/08/a-comunicacao-como-fomentadora-de.html. Acesso em: 8 fev. 2025.

TRILICONE, Matheus. Doenças neuromusculares. Blog Matheus Trilicone Neurologia. Disponível em: https://blog.matheustriliconeurologia.com.br/doencas-neuromusculares/. Acesso em: 9 fev. 2025.

UGUSTA, Flávia. Dificuldades de comunicação em idosos. Flávia Augusta Fono, 2023. Disponível em: https://flaviaaugustafono.com.br/dificuldades-de-comunicacao-em-idosos/. Acesso em: 10 fev. 2025.

SILVA, José. Comunicação para ELA utilizando Arduino. Embarcados. Disponível em: https://embarcados.com.br/comunicacao-para-ela-utilizando-arduino/. Acesso em: 11 fev. 2025.