Residência em Sistemas Aluno: Vinícius da Silva Barreto dos Santos

Unidade 4 – Microcontroladores

Instrutores: Cleia Libarino, Ricardo Prates e Wilton Lacerda

Projeto Final – Videogame Educativo

1. Escopo do Projeto

Este projeto final trata de uma implementação feita na placa Bitdoglab, que é microcontrolada pelo microcontrolador Raspberry Pi Pico W. O projeto consiste em um mini videogame educativo.

De caráter simplista, a proposta é que o console torne-se uma ferramenta de identificação de notas musicais (mini game 1) e memorização (mini game 2). Dessa forma, buscando contribuir para o desenvolvimento da percepção de sons e padrões, principalmente para crianças, que são o principal público-alvo de jogos e deveriam ser encorajadas a aprender um instrumento nesta idade, a fim de desenvolver o raciocínio e a escuta ativa.

O mini game 1, batizado de "Ouvido Musical" possui 4 músicas que tocarão para o usuário duas vezes. Na primeira vez será essencial que o usuário se atente às notas tocadas e, na segunda vez, deverá selecionar as notas musicais faltantes a cada pausa da música. Serão exibidas 3 opções de notas para cada música. Caso o usuário acerte as três notas da música, avançará para a música seguinte. Do contrário, repetirá o processo. Ao final do processo, retornará para o menu inicial.

O mini game 2, batizado de "Jogo da Memória", possui 10 sequências que serão exibidas na matriz de LEDs. A ideia é que o usuário memorize a sequência correta e repita quando for a sua vez. É essencial que o usuário repita a sequência de forma correta, a fim de avançar no jogo. Cada avanço gera 1 ponto para o usuário. Caso o usuário erre alguma sequência, retornará ao nível 1 e sua pontuação será zerada. Ao chegar ao último nível, o usuário formará 10 pontos e retornará para o menu inicial.

O primeiro mini game é uma inspiração do mini game "Beats Me" do jogo "SpongeBob Squarepants: Light, Camera, Panos". O game consistia em repetir os padrões de ritmo do personagem Squilliam no instrumento do seu personagem.

O segundo mini game se assimila muito com o game "Genius", muito famoso nos anos 90, onde precisávamos repetir as cores sequencialmente, a fim de avançar. Conforme avançávamos, o jogo se tornava mais complexo por exibir mais cores em uma única sequência.

2. Especificação de Hardware

Para a execução deste projeto foram utilizados diversos recursos da placa Bitdoglab. Os principais recursos utilizados foram: GPIOs, PIO, ADC e a comunicação I2C. A tabela a seguir descreve a função de cada componente do projeto:

Bloco	Função
Microcontrolador (RP2040)	Responsável pelo processamento dos jogos, controle dos periféricos e execução das lógicas de interação.

Display OLED SSD1306	Exibe as informações dos jogos, como menu, mensagens e seleção de notas. Comunicação via I2C.
Joystick Analógico	Controla a navegação entre as opções do jogo e a seleção de notas musicais. Lido via ADC.
Botões (A e B)	Interação do usuário, com função específica para cada botão. Leitura via GPIO com pull-up interno.
Buzzer	Produz os sons do jogo musical e efeitos sonoros no jogo da memória. PWM utilizado diretamente na função do buzzer
Matriz de LEDs WS2812	Exibe padrões visuais no jogo da memória. Controlada via PIO para temporização precisa.

Abaixo listo as configurações para cada um dos blocos utilizados neste projeto.

Microcontrolador RP 2040 – Pi Pico W: Operação em 3,3V e Clock de 128 MHz. Foram utilizadas as seguintes interfaces: GPIOs para os botões e buzzers, I2C para o display OLED, ADC para o joystick analógico e PIO para a matriz de LEDs.

Display OLED SSD1306: Foi utilizado o protocolo I2C e, para tal, é necessário setar algumas configurações. Os bits padrões deste protocolo, SDA (GPIO 14) e SCL (GPIO15) . Após, configurar o display com a inclusão do seu header *ssd1306.h*. Para inicializar o display utilizou-se: ssd1306_init(&ssd, WIDTH, HEIGHT, false, endereco, I2C_PORT); .

Para enviar os dados para o display utilizou-se a função ssd1306_send_data(&ssd); .

Joystick Analógico: Utilizou-se a conversão ADC. Para tal, há 2 encoders no analógico. Um para o eixo X (chamado de VRX e com o correspondente GPIO26 no canal 1 do ADC) e outro para o eixo Y (chamado de VRY e com o correspondente GPIO 27 no canal 0 do ADC). O ADC é configurado com o comando adc_init(); , os pinos são ativados com o adc_gpio_init(); e, por fim, a leitura do ADC é feita por adc_read();.

Botões A e B: São configurados como simples entradas digitais. São 2 botões, BOTAO_A (GPIO5) e BOTAO_B(GPIO6). Necessário estabelecer uma sequência de configuração, a partir da sequência das funções: gpio_init (); , gpio_set_dir(); , e gpio_pull_up (); . Esta última função define a entrada no pull up, ou seja, enquanto não for acionada, a entrada estará em nível lógico alto. Ao ser acionado, vai para nível lógico 0. Necessário implementar um debouncing no botão, que foi feito com a função time_us_64 ();.

Buzzer: O buzzer é um gpio, definido como saída. Consiste em um gerador de frequência para produção de sons. Temos 2 buzzers. O buzzer utilizado no primeiro mini game trata-se do BUZZER_PIN (GPIO21) e o BUZZER (GPIO10). O primeiro é utilizado no mini game musical e o segundo é utilizado no mini game do jogo da memória. É inicializado pelas funções gpio_init (); e gpio_set_dir();. As frequências são controladas via play_tone( );.

Matriz de LEDs WS2812: Controlada via Pio (Programmable I/o) para manter a precisão na temporização dos pulsos. É configurada a definida a partir de MATRIZ (GPIO7). O código PIO é carregado a partir da função pio_add_program (); , e a máquina de estados é configurada por nome_do_projeto_program_init (). Os LEDs são atualizados pela função put_leds();.

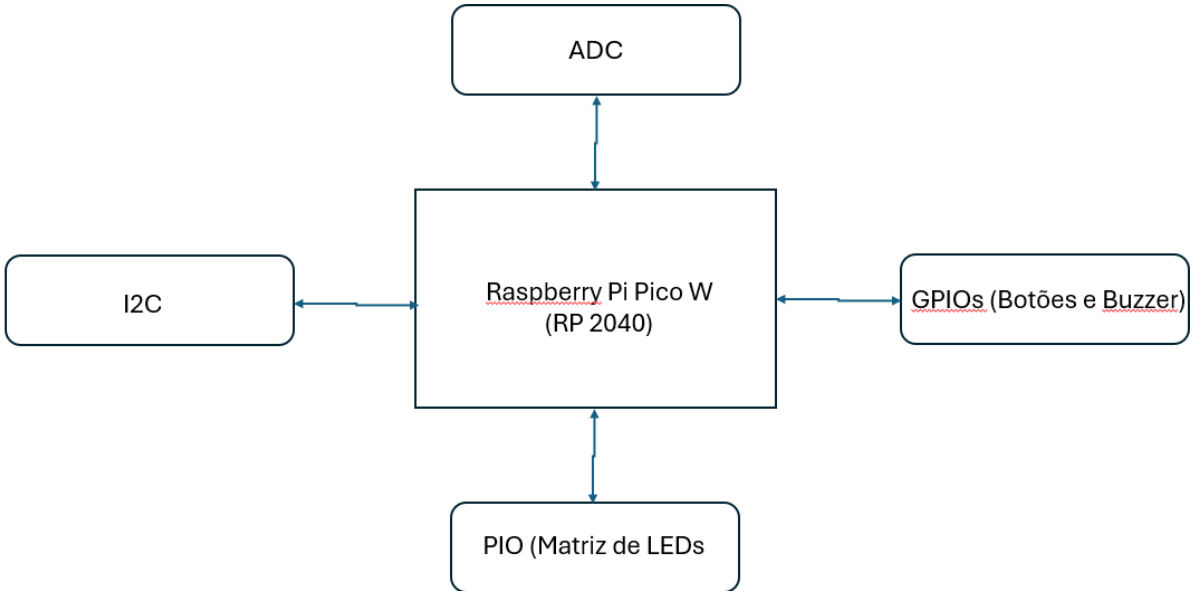

Diagrama de Bloco

Comandos e registros utilizados:

Para a configuração do Clock utilizou-se a seguinte definição no código:

- bool ok = set_sys_clock_khz(128000, false);

Para a configuração do protocolo I2C para o display OLED utilizou-se:

- 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);

Principais Registros Utilizados para Controle

Recurso	Registrador	Comando Usado
Leitura ADC (Joystick)	NUM_ADC_CHANNELS, ADC_CS_AINSEL_LSB, ADC_CS_AINSEL_BITS, ADC_CS_START_ONCE_BITS e ADC_CS_READY_BITS	adc_select_input(0); ,adc_read();
Controle do Buzzer (PWM)	PWM_CHAN_LEVEL	play_tone(BUZZER, freq, duration);
Controle da Matriz WS2812 (PIO)	PIO_SMx_SHIFTCTRL	put_leds();
Controle dos Botões (GPIO)	GPIO_IN	gpio_get(BOTAO_A);
Controle do Display SSD1306 (I2C)	I2C_CR1, I2C_DR	ssd1306_send_data();

Abaixo, as definições dos pinos:

#define VRX_PIN 26

#define VRY_PIN 27

#define BOTAO_A 5

#define BOTAO_B 6

#define MATRIZ 7

#define BUZZER 10

#define I2C_PORT i2c1

#define I2C_SDA 14

#define I2C_SCL 15

#define endereco 0x3C

#define BUZZER_PIN 21

3. Especificações de Firmware

Descrição de Funcionalidades

Interface do Usuário: Exibe informações no display OLED e responde aos botões físicos.

Controle dos Jogos: Gerencia a lógica do jogo da memória e do jogo musical.

Módulos de Hardware: Controla a interação com ADC (joystick), PIO (LEDs), PWM (buzzer), GPIO (botões) e I2C (display OLED).

Comunicação com Periféricos: Utiliza o Pico SDK para acessar os recursos do RP2040.

Firmware RP2040: Contém o loop principal e inicializa os periféricos do sistema.

Principais pontos:

Interface do Usuário: Exibe informações no display OLED e responde aos botões físicos.

Controle dos Jogos: Gerencia a lógica do jogo da memória e do jogo musical.

Módulos de Hardware: Controla a interação com ADC (joystick), PIO (LEDs), PWM (buzzer), GPIO (botões) e I2C (display OLED).

Comunicação com Periféricos: Utiliza o Pico SDK para acessar os recursos do RP2040.

Firmware RP2040: Contém o loop principal e inicializa os periféricos do sistema.

Inicialização do software:

Configuração do Clock: Define a frequência do sistema para 128MHz.

Inicialização do ADC: Ativa a leitura do joystick.

Configuração dos GPIOs: Define os pinos dos botões e do buzzer.

Configuração do PIO: Configura a comunicação com a matriz de LEDs.

Inicialização do Display OLED: Configura o barramento I2C e limpa a tela.

Entrada no Menu: Exibe a interface inicial para seleção do jogo.

Protocolo de Comunicação:

I2C: Usado para comunicação entre o RP2040 e o display OLED.

PIO: Utilizado para controlar a matriz de LEDs 5x5.

PWM: Controla o buzzer para a geração de notas musicais.

Formato do Pacote de Dados

I2C (Display OLED):

- Endereço do dispositivo: 0x3C
- Pacote: {Start Bit} + {Endereço} + {Comando/Dados} + {Stop Bit}

PIO (Matriz de LEDs):

- 25 bits por ciclo (1 bit por LED)
- Sequência de atualização baseada em *frames*

PWM (Buzzer):

- Frequência ajustada por PWM_CH0_DIV
- Duração da nota controlada por sleep_ms()

Definição das principais variáveis

Variável	Tipo	Função
musica_atual	int	Armazena o índice da música atual
nivel	uint	Nível do jogo da memória
pontuacao	int	Armazena a pontuação do jogo da memória
tempo_anterior	uint64_t	Controla debounce e temporizações
opcao	int	Define qual jogo foi selecionado
ssd	ssd1306_t	Estrutura para controle do display OLED

Para facilidade do primeiro minigame, todas as respostas são as 3 primeiras notas de cada música. Para o segundo minigame, necessário observar no código. As matrizes contendo as

sequências do segundo minigame estão todas declaradas no header “memoria.h”. O header “musica.h” lista as músicas e realiza os tratamentos propostos, como tocar a nota ou pausar na nota correspondente.

Para avançar no jogo musical, é necessário acertar as 3 notas. Onde tiver REST significa uma pausa musical. Se errar uma nota, repete a música toda.

O mesmo para o jogo de memorização. Necessário repetir a sequência correta, a fim de avançar no código. Caso erre, sua pontuação é zerada e você retorna para o nível 1 do jogo.

4. Execução do Projeto

Como o projeto se trata de um mini console de jogos, as depurações envolveram diretamente a placa, ou seja, foram adicionados poucas flags no código. As flags foram removidas assim que apurei que estava tudo correto e perfeitamente funcional.

No mini game do jogo da memória, o principal ponto de depuração foi o avanço da pontuação ou zerar, a depender do que ocorra.

No mini game musical, a maior dificuldade foi fazer com que retornasse para a primeira música, em caso de erro.

O projeto foi interessante, desafiador e complexo. Todavia, busquei me desafiar e fazer algo que acredito combinar comigo e me trazer algumas lembranças. Ao longo deste projeto pude perceber quão poderosa é a Raspberry Pi Pico W frente a outros microcontroladores que já utilizei ao longo da vida.

Quanto à confiabilidade do projeto, foi testado diversas vezes e está funcional, de acordo com o que eu esperei e projetei.

5. Referências:

Datasheet da Raspberry Pi Pico W

DAMAS, Luis. Linguagem C. 10ª Edição. Rio de Janeiro: LTC,2014.