UNIVERSIDADE FEDERAL DO VALE DO SÃO FRANCISCO

LEITOR ANALÓGICO DIGITAL COM EXIBIÇÃO NO MONITOR SERIAL ATRAVÉS I2C PARA VALORES DE TENSÃO

Engenharia Elétrica

Alex dos Santos Bomfim, Alexandre Fernandes das Neves Junior e Erick da Silva Sousa

Universidade Federal do Vale do São Francisco (UNIVASF)

Resumo – Nesta atividade foi desenvolvido um voltímetro digital embarcado usando o microcontrolador RP2040 (Raspberry Pi Pico W) e a placa BitDogLab. O joystick forneceu tensão analógica que foi convertida pelo ADC de 12 bits do RP2040 em valores de 0–4095, correspondentes a 0–3,3 V. Esses valores foram transmitidos via barramento I2C ao monitor serial da BitDogLab, exibindo a leitura em tempo real.

Palavras-Chave – RP2040; I2C; Raspberry Pi Pico W.

ANALOG DIGITAL READER WITH DISPLAY ON SERIAL MONITOR THROUGH I2C FOR VOLTAGE VALUES

Abstract – This activity developed an embedded digital voltmeter using the RP2040 microcontroller (Raspberry Pi Pico W) and the BitDogLab board. The joystick provided analog voltage, which was converted by the RP2040's 12-bit ADC into values ranging from 0–4095V, corresponding to 0–3.3V. These values were transmitted via the I2C bus to the BitDogLab serial monitor, displaying the reading in real time.

Keywords – RP2040; I2C; Raspberry Pi Pico W.

I. INTRODUÇÃO

Neste projeto, foi utilizado o Conversor Analógico-Digital (ADC) do RP2040 para ler as tensões proporcionadas por um joystick integrado à placa BitDogLab e apresentá-las em tempo real num monitor serial I2C. A Raspberry Pi Pico W, com seu ADC de 12 bits, converte valores de 0–4095 em tensões de 0–3,3 V, enquanto o barramento I2C estabelece uma interface eficiente para exibir essas leituras. O resultado é um voltímetro digital embarcado, que demonstra tanto a precisão da conversão analógica quanto a confiabilidade do protocolo I2C em sistemas de monitoração embarcados.

II. FUNDAMENTAÇÃO TEÓRICA

O I2C (Inter-Integrated Circuit) é um protocolo de comunicação serial síncrona de dois fios, SDA (dados) e SCL (clock), originalmente desenvolvido pela Philips (hoje NXP) em 1982. Em sistemas embarcados, incluindo o RP2040 da Raspberry Pi Pico W, ele permite que múltiplos dispositivos (mestres e escravos) compartilhem o

mesmo barramento, cada um identificado por endereços únicos de 7 ou 10 bits.

Arquitetura Física

As linhas SDA e SCL são de coletor aberto (open-drain), exigindo resistores de pull-up para defini-las em nível alto.

Qualquer dispositivo no barramento pode puxar a linha para nível baixo para transmitir “0”; ao liberar, o resistor de pull-up leva a linha a nível alto “1”.

Protocolo de Transmissão

Start: mestre puxa SDA de alto para baixo enquanto SCL está alto.

Endereçamento: mestre envia o endereço de 7 bits seguido de um bit R/W.

Acknowledge (ACK/NACK): receptor baixa SDA no nono pulso de clock para indicar recepção.

Data: cada byte transferido é seguido de um ACK.

Stop: mestre libera SDA (baixo→alto) enquanto SCL está alto, encerrando a transferência.

Velocidades de Operação

Modos Standard com 100 kHz, Fast 400 kHz, Fast-Plus 1 MHz e High-Speed 3,4 MHz. Para o BitDogLab, normalmente se usa até Fast Mode, suficiente para atualizar displays e monitores seriais I2C em tempo real.

I2C no RP2040 e BitDogLab

O RP2040 expõe módulos I2C0 e I2C1, cada um mapeado a dois pinos GPIO configuráveis como SDA e SCL.

No BitDogLab, o barramento I2C está roteado a um conector de monitor serial, onde podemos exibir valores lidos pelo ADC do RP2040, transformando-o em um voltímetro digital: Fazendo 0 = 0V e 4095 = 3,3V (Tensão máxima permitida para entrada).

A leitura ADC é convertida em tensão e enviada sobre I2C para o display, demonstrando a integração de leitura analógica e comunicação serial.

Potenciômetro (Joystick)

O joystick funciona como um divisor de tensão, em que cada eixo de movimento (X e Y) altera a resistência de um potenciômetro. No ponto central, considerando uma tensão de referência de 3,3 V, a tensão correspondente será a metade desse valor, ou seja, 1,65 V. Nos extremos (mínimo e máximo), os valores de tensão serão, respectivamente, 0 V e 3,3 V.

Portanto, para implementar a lógica no código, é essencial entender o funcionamento do joystick e como o microcontrolador interpreta essa tensão. No caso, o microcontrolador converte a tensão em um valor digital entre 0 e 4096. Assim, utilizaremos estruturas condicionais (if e else) para exibir na tela OLED a tensão correspondente à posição do joystick. Por exemplo: joystick totalmente para cima → 4096 → 3,3 V.

III. RESULTADOS OBTIDOS

Utilizando as bibliotecas: **hardware/i2c.h** (responsável pelo funcionamento da i2c); **inc/ssd1306.h**(responsável por controlar o display oled); **inc/font.h**(responsável pelas fontes e figuras imprimidas no display); **hardware/adc.h** (Ele é

responsável por controlar todo o periférico ADC), e seguindo a logica das estruturas condicionais, podemos formular um código da seguinte maneira:

CODIGO PARA EXIBIÇÃO NO MONITOR SERIAL ATRAVÉS I2C PARA VALORES DE TENSÃO

```
#include <stdlib.h>
#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/i2c.h"
#include "inc/ssd1306.h"
#include "inc/font.h"
#include "hardware/adc.h"

#define I2C_PORT i2c1
#define I2C_SDA 14
#define I2C_SCL 15
#define endereco 0x3C

float v_max = 3.3, v_out = 0.0,
    v_min = 0.0, js_max = 4096.0,
    js_in = 0.0, js_min = 0.0;

char var_string[20];



int main()
{
 stdio_init_all();
 sleep_ms(200);

 //Configuração do canal AD
 adc_init();
 adc_gpio_init(26);
 adc_select_input(0);

 // I2C Initialisation. Using it at 400Khz.
 i2c_init(I2C_PORT, 400 * 1000);

 gpio_set_function(I2C_SDA, GPIO_FUNC_I2C);
// Set the GPIO pin function to I2C
 gpio_set_function(I2C_SCL, GPIO_FUNC_I2C);
// Set the GPIO pin function to I2C
 gpio_pull_up(I2C_SDA); // Pull up the data line
 gpio_pull_up(I2C_SCL); // Pull up the clock line
 ssd1306_t ssd; // Inicializa a estrutura do
display
 ssd1306_init(&ssd, WIDTH, HEIGHT, false,
endereco, I2C_PORT); // Inicializa o display
 ssd1306_config(&ssd); // Configura o display
 ssd1306_send_data(&ssd); // Envia os dados
para o display

 // Limpa o display. O display inicia com todos
os pixels apagados.
 ssd1306_fill(&ssd, false);
 ssd1306_send_data(&ssd);

 bool cor = true;
 while (true)
 {
  js_in = adc_read();

  if (js_in < js_min)
   {
    v_out = v_min;
   }
   else if(js_in > js_max)
   {
    v_out = v_max;
   }
   else
```

```
    {
      v_out = ((js_in - js_min)*(v_max - v_min))/(js_max - js_min) + v_min;
    }


   sprintf(var_string, "%.2f", v_out);


   // Atualiza o conteúdo do display
   ssd1306_fill(&ssd, !cor); // Limpa o display
   ssd1306_rect(&ssd, 0, 0, 128, 63, !cor, !cor);
   ssd1306_draw_string(&ssd, "UNIVASF   ELET", 8, 10);
   ssd1306_draw_string(&ssd, "MICRO", 40, 20);
   ssd1306_draw_string(&ssd, var_string, 40, 45);
   ssd1306_pixel(&ssd, 49, 51, cor);
   ssd1306_draw_string(&ssd, "V", 75, 45);
   ssd1306_send_data(&ssd); // Atualiza o display

   sleep_ms(100);
  }
}
```

O código lê um sinal analógico “adcread()” , converte esse sinal em tensão

```
if (js_in < js_min)
    {
     v_out = v_min;
    }
    else if(js_in > js_max)
    {
     v_out = v_max;
    }
    else
    {
      v_out = ((js_in - js_min)*(v_max - v_min))/(js_max - js_min) + v_min;
    }
```

E imprime o valor em um display OLED controlado pelo driver SSD1306

```
sprintf(var_string, "%.2f", v_out);


   // Atualiza o conteúdo do display
   ssd1306_fill(&ssd, !cor); // Limpa o display
   ssd1306_rect(&ssd, 0, 0, 128, 63, !cor, !cor);
   ssd1306_draw_string(&ssd, "UNIVASF   ELET", 8, 10);
   ssd1306_draw_string(&ssd, "MICRO", 40, 20);
   ssd1306_draw_string(&ssd, var_string, 40, 45);
   ssd1306_pixel(&ssd, 49, 51, cor);
   ssd1306_draw_string(&ssd, "V", 75, 45);
   ssd1306_send_data(&ssd); // Atualiza o display
```

esse driver é controlado pelo barramento I2C e configurado da seguinte forma:

```
#define I2C_PORT i2c1
#define I2C_SDA 14
#define I2C_SCL 15
#define endereco 0x3C

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);
```

IV.CONCLUSÃO

A implementação do voltímetro digital reforçou a capacidade do RP2040 de capturar sinais analógicos com alta resolução e transmiti-los via I2C, evidenciando a sinergia entre ADC e comunicação serial em aplicações reais. Aprendemos a calibrar leituras e configurar o barramento I2C de forma funcional. Como desdobramentos futuros, pode-se adicionar múltiplos canais de medição, implementar filtragem digital no firmware ou integrar registro de dados via Wi-Fi, explorando todo o potencial de conectividade e processamento do RP2040.

REFERÊNCIAS

[1] RASPBERRY PI LTD. RP2040 Datasheet.[S.1.]: Raspberry Pi Ltd., 2020.

[2] RASPBERRY PI LTD. Raspberry Pi Pico W Datasheet.[S.1.]: Raspberry Pi Ltd., 2022.

[3] RASPBERRY PI LTD. Pico C/C++SDK – Raspberry Pi Documentation. [S.1.]: Raspberry Pi Ltd., 2023.

[4] BITDOGLAB. BitDogLab RP2040 Datasheet (V 5.3.1). [S.1.]: BitDogLab, [s.d.].