



UPDF
WWW.UPDF.COM



embarca
tech

Residência
Tecnológica
em Sistemas
Embarcados

Práticas de Portas de Entrada e Saída – GPIO

Unidade 4 | Capítulo 2

Tiago Façanha

Executores:



INSTITUTO FEDERAL
Piauí



INSTITUTO FEDERAL
Rio Grande do Norte



INSTITUTO FEDERAL
Maranhão



INSTITUTO FEDERAL
Ceará



INSTITUTO
HARDWARE BR

Coordenação:



Iniciativa:

GOVERNO FEDERAL
MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA
E INOVAÇÃO
BRASIL
UNIÃO E RECONSTRUÇÃO



Sumário

1	Boas-vindas e Introdução.....	3
2	Objetivos desta Unidade.....	4
3	Resumo da unidade anterior e exemplos práticos de comunicação dos microcontroladores.....	5
4	Bounce	7
4.1.	O que é o "Bounce"?	7
4.2.	Consequências do "Bounce"	8
4.3.	Soluções	8
5.	Conclusão	10
	REFERÊNCIAS.....	11

Unidade 4

Microcontroladores

Capítulo 2

OBJ 1: Utilizar o módulo de entrada e saída para controle de LED e leitura de chave.

OBJ 2: Utilizar o módulo de entrada e saída para controle de cor de LEDs com dois botões.

OBJ 3: Utilizar o módulo de entrada e saída para implementar a leitura de um teclado matricial

1. BOAS-VINDAS E INTRODUÇÃO

Exemplo: “Olá, estudantes! Sejam bem-vindos à nossa aula sobre [Tema da Aula].”

Neste eBook, vamos explorar o universo dos GPIOs (General Purpose Input/Output) utilizando o microcontrolador Raspberry Pi Pico e a plataforma educacional BitDogLab. Este material foi desenvolvido para expandir suas habilidades em eletrônica e programação de microcontroladores.

Os GPIOs são fundamentais para conectar o mundo digital ao mundo físico. Eles permitem que o microcontrolador interaja com diversos dispositivos e sensores, como LEDs, botões, teclados matriciais, motores e muitos outros componentes. A partir do controle desses pinos, é possível desenvolver projetos de diferentes níveis de complexidade, desde o acionamento de um simples LED até a criação de sistemas interativos completos.

A plataforma BitDogLab[1], como já aprendemos, é baseada no Raspberry Pi Pico[2] e foi concebida para facilitar o aprendizado prático. Com uma vasta gama de componentes integrados e uma

interface amigável, ela oferece um ambiente ideal para explorar o potencial dos GPIOs. Neste eBook, utilizaremos a plataforma wokwi e BitDogLab para realizar uma série de experimentos e projetos práticos, permitindo que você aprenda enquanto coloca a mão na massa.

Ao longo deste material, apresentaremos conceitos teóricos essenciais, seguidos de exemplos práticos que demonstram a aplicação desses conceitos. Você aprenderá como configurar e utilizar os GPIOs para controlar LEDs, ler o estado de botões, criar sequências de controle com LEDs RGB, e até mesmo implementar a leitura de um teclado matricial. Tudo isso de forma didática e detalhada, para que você possa acompanhar cada passo do processo e desenvolver suas próprias soluções.

Além disso, abordaremos boas práticas de desenvolvimento, como o tratamento de ruídos em botões mecânicos, conhecido como “debouncing”, e discutiremos técnicas para evitar problemas comuns ao trabalhar com sinais digitais. Ao final deste eBook, você terá uma compreensão sólida sobre o uso dos GPIOs e estará preparado para criar seus próprios projetos de sistemas embarcados com confiança.

Seja bem-vindo a essa jornada de aprendizado e descoberta. Prepare seus componentes, conecte seu Raspberry Pi Pico e vamos começar a explorar as infinitas possibilidades dos GPIOs com a plataforma BitDogLab!

2 Objetivos desta Unidade

Ao final desta aula, espera-se que você compreenda a funcionalidade de cada pino GPIO do microcontrolador Raspberry Pi Pico, sendo capaz de diferenciar suas aplicações como entradas e saídas digitais, e configurá-los de forma adequada para interagir com uma variedade de componentes eletrônicos. Além disso, você estará apto a implementar e utilizar teclados matriciais e outros dispositivos de entrada para capturar comandos do usuário, desenvolvendo interfaces interativas e sistemas de controle. Serão desenvolvidas habilidades críticas para a resolução de problemas típicos de sistemas embarcados, como o tratamento de ruídos gerados por contatos mecânicos, aplicando técnicas de “debouncing”[3] tanto por meio de soluções de hardware quanto por algoritmos de software,

garantindo a robustez e a confiabilidade das leituras de sinais.

3 Resumo da unidade anterior e exemplos práticos de comunicação dos microcontroladores

Nos capítulos anteriores, foram abordados os **conceitos fundamentais do desenvolvimento** com microcontroladores, com foco na placa de desenvolvimento BitDogLab, baseada no Raspberry Pi Pico W. Os conteúdos cobriram as **principais características** desse microcontrolador, incluindo a estrutura de hardware, que integra processadores, memória e uma variedade de periféricos para controle e comunicação. Também foi apresentada a organização das bibliotecas e kits de desenvolvimento de software (SDKs) fornecidos pelos fabricantes, que simplificam a programação dos microcontroladores e permitem o uso eficiente de suas funcionalidades.

O microcontrolador RP2040, presente na Raspberry Pi Pico W, foi explorado em detalhes, destacando suas capacidades e interfaces, como GPIO, UART, I2C e SPI [4], que possibilitam a **interação** com diversos componentes externos. Foram discutidas também as vantagens e desafios do uso de microcontroladores em sistemas embarcados, incluindo suas limitações de processamento e memória, que exigem uma otimização cuidadosa do código e uma gestão eficiente dos recursos para superar as restrições impostas por esses dispositivos.

Além disso, foi apresentada a plataforma BitDogLab, que oferece um ambiente de **aprendizado prático** com uma variedade de periféricos integrados, como displays, botões e matriz de LEDs, além de conectividade sem fio. Essa plataforma foi utilizada para ilustrar exemplos práticos e experimentos interativos que auxiliam na consolidação dos conceitos teóricos. A combinação desses recursos com o SDK do Raspberry Pi Pico fornece uma base sólida para o desenvolvimento de aplicações complexas, preparando os alunos para lidar com projetos reais em sistemas embarcados.

Hoje, vamos construir sobre esse conhecimento adquirido e explorar o **uso prático** dos GPIOs com o microcontrolador Raspberry Pi Pico, utilizando a plataforma BitDogLab. Neste material, abordaremos novos conteúdos que incluem a configuração de pinos como entradas e saídas

digitais, a implementação de controles de LEDs e leitura de botões, e o desenvolvimento de sistemas interativos utilizando teclados matriciais.

Este tópico é importante porque permite que você compreenda como os microcontroladores se comunicam com o mundo físico, transformando sinais digitais em ações concretas e vice-versa. A capacidade de controlar dispositivos eletrônicos e de capturar comandos de usuários é essencial para a criação de sistemas embarcados funcionais, e o domínio dessas habilidades é fundamental para o desenvolvimento de soluções inovadoras e eficientes em uma ampla gama de aplicações, desde projetos educacionais até implementações industriais.

Nesta seção, vamos explorar três exemplos práticos que ilustram o uso dos GPIOs no desenvolvimento de projetos com o Raspberry Pi Pico, cada um abordado em um subtópico específico deste eBook.

/INÍCIO ATENÇÃO/

Cada exemplo apresentará diagramas, códigos e explicações detalhadas, servindo como referência prática para o desenvolvimento de seus próprios projetos com a plataforma BitDogLab.

/FIM ATENÇÃO/

O primeiro exemplo, “Controle de LED e Leitura de Botão”, fornece uma base para compreender como interagir com componentes eletrônicos básicos.

- [Clique aqui para ir para o exemplo 1 \(página 13\)](#)

O segundo exemplo, “Controle de LEDs RGB e Leitura de Dois Botões”, demonstra como manipular múltiplas saídas visuais com base em diferentes condições de entrada.

- [Clique aqui para ir para o exemplo 2 \(página 17\)](#)

Por fim, em “Leitura de Teclado Matricial”, discutiremos como capturar comandos de forma eficiente utilizando um número reduzido de pinos, viabilizando a criação de interfaces de usuário mais sofisticadas.

• [Clique aqui para ir para o exemplo 3 \(página 23\)](#)

4 Bounce

Antes de começarmos a implementar nossos projetos com entradas digitais, como botões, é fundamental compreender um fenômeno comum em sistemas eletrônicos: o "bounce". O "bounce" ocorre devido às características mecânicas dos botões e interruptores e pode afetar a precisão e confiabilidade das leituras de estado em sistemas embarcados.

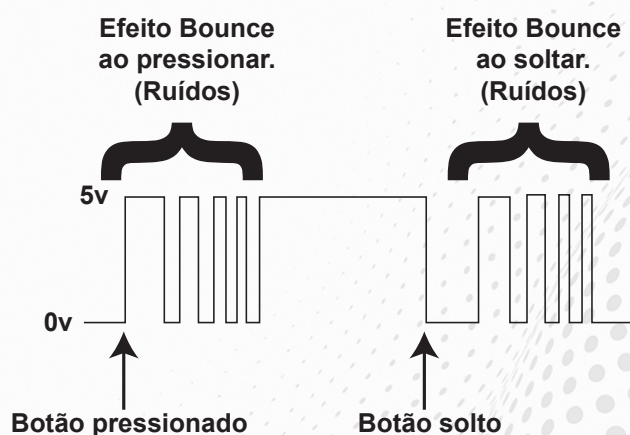
4.1. O que é o "Bounce"?

O "bounce", ou "rebote", é um efeito indesejado que acontece quando um botão é pressionado ou liberado. Em um mundo ideal, ao pressionar um botão, o contato seria estabelecido de forma instantânea e limpa, resultando em uma única transição de estado de aberto para fechado (ou vice-versa). No entanto, na realidade, ao pressionar ou soltar um botão, o contato mecânico não é feito de maneira uniforme, e ocorre uma série de pequenos "saltos" ou oscilações antes de o contato se estabilizar definitivamente. **Essas oscilações são interpretadas pelo microcontrolador como múltiplos sinais de transição, gerando leituras incorretas e potencialmente acionando funções indesejadas várias vezes.**

A Figura 3.4.1 mostra como o sinal elétrico oscila entre os estados lógico alto e baixo quando o botão é pressionado. O gráfico apresentado na figura evidencia que, durante uma única pressão do botão, o sinal pode alternar várias vezes entre "1" e "0" antes de se estabilizar em um dos estados. Esse comportamento é devido à elasticidade dos materiais e à instabilidade do contato mecânico, que se comporta como uma mola, causando essas múltiplas transições em um curto espaço de tempo.

Fig. 1 - ilustração do efeito bounce e possíveis soluções. Efeito que ocorre ao pressionar o botão 1 vez.

Fonte: imagem do autor.



4.2. Consequências do "Bounce"

O "bounce" pode causar vários problemas em projetos de sistemas embarcados, especialmente quando utilizamos botões para interagir com o microcontrolador. Cada oscilação pode ser interpretada como uma ativação ou desativação do botão, levando o sistema a detectar múltiplos pressionamentos quando, na verdade, ocorreu apenas um. Isso pode resultar em acionamentos indesejados, problemas em contadores de eventos, e falhas em sistemas que dependem de precisão, como controles de menu ou interfaces de usuário.

4.3. Soluções

Para amenizar os efeitos do "bounce", existem duas abordagens principais:

1. Solução por Hardware:

- Utilização de componentes de alta qualidade que minimizam o efeito de "bounce".
- Implementação de filtros RC (resistor-capacitor) para suavizar o sinal, eliminando as oscilações indesejadas.

2. Solução por Software:

- Implementação de técnicas de "debouncing" por software, onde se espera um curto período (alguns milissegundos) após a detecção de uma mudança de estado antes de considerar o botão pressionado ou liberado. Este atraso permite que o sinal se estabilize antes de ser lido novamente.
- A técnica mais comum envolve a utilização de temporizadores ou contadores no código, que garantem que a mudança de estado só será registrada se o sinal permanecer constante por um determinado período.

Compreender e tratar o "bounce" é essencial para qualquer projeto que envolva interação com botões ou interruptores. A correta implementação de técnicas de "debouncing", seja por hardware ou software, garante que seu sistema embarcado opere de forma precisa e confiável, evitando comportamentos indesejados e falhas na interface com o usuário.

/INICIO EM SINTESE/

Revisando, aprendemos que o uso dos módulos de entrada e saída (GPIOs) do Raspberry Pi Pico permite a criação de uma variedade de interações e controles em sistemas embarcados. No primeiro exemplo, onde utilizamos um pino de saída para controlar o acendimento de um LED e um pino de entrada para ler o estado de um botão. Esse exemplo básico forneceu a base para compreender a configuração e manipulação dos GPIOs, possibilitando a construção de circuitos simples de controle e resposta.

Em seguida, avançamos para o controle de cores de LEDs com dois botões, explorando o uso simultâneo de múltiplos pinos de entrada e saída. Aprendemos a utilizar três LEDs de diferentes cores (vermelho, verde e azul) e a alternar suas combinações com base no estado dos botões. Esse exemplo demonstrou como podemos criar interfaces visuais mais interativas e dinâmicas, utilizando técnicas básicas de lógica digital para manipular múltiplos sinais de forma simultânea.

Por fim, implementamos a leitura de um teclado matricial 4x4, onde utilizamos oito pinos GPIO para capturar as interações de até 16 teclas. Esse exemplo destacou a eficiência do uso de teclados matriciais para capturar comandos complexos com um número reduzido de pinos, permitindo o desenvolvimento de interfaces de usuário mais sofisticadas e funcionais.

Além disso, discutimos o conceito de “bounce”, um fenômeno comum em botões mecânicos, onde pequenas oscilações no sinal de entrada podem levar a leituras incorretas. Abordamos a importância de mitigar esse efeito através de técnicas de “debouncing” por hardware ou software, garantindo a precisão e a confiabilidade dos sistemas embarcados que dependem de entradas digitais. Esses conceitos e práticas fornecem a base para o desenvolvimento de projetos mais robustos e interativos, permitindo explorar todo o potencial dos GPIOs no controle e na comunicação com o mundo físico.

/FIM EM SINTESE/

Não se esqueça de revisar todo o material estudado nesta aula e colocar em prática os conceitos aprendidos nos exemplos e exercícios propostos.

/NA PRÁTICA/

Complete as atividades práticas disponíveis na plataforma, pois elas são essenciais para consolidar o seu entendimento sobre o uso dos GPIOs. Experimente criar seus próprios projetos, testando diferentes combinações de entradas e saídas. Quanto mais você praticar, mais domínio terá sobre esses conceitos. Vamos em frente!

/NA PRÁTICA/

6. CONCLUSÃO

Concluimos esta aula com uma compreensão sólida sobre como utilizar os módulos de entrada e saída (GPIOs) do Raspberry Pi Pico para controlar LEDs e capturar interações de usuários através de botões e teclados matriciais. Exploramos conceitos fundamentais, como a configuração de pinos como entradas e saídas, e aplicamos esse conhecimento em exemplos práticos que mostraram como controlar LEDs simples, múltiplos LEDs com botões, e ler um teclado matricial para capturar comandos mais complexos.

No próximo capítulo, avançaremos para práticas mais desafiadoras de controle de GPIOs com temporização. Abordaremos a implementação de diferentes exemplos que combinam controle de hardware e rotinas de atraso, como o controle temporizado de um semáforo utilizando push-buttons, o uso de displays de 7 segmentos para exibir informações com temporização e multiplexação, e o projeto de um sistema de controle para um cofre de 4 dígitos. Esses novos desafios irão reforçar a importância de técnicas de temporização e gerenciamento eficiente de recursos, preparando-o para desenvolver aplicações embarcadas ainda mais complexas e funcionais. Prepare-se para explorar o mundo dos atrasos programados e como eles podem ser utilizados para criar sistemas de controle precisos e interativos.

REFERENCIAS

- [1] BITDOGLAB. BitDogLab. Disponível em: <https://github.com/BitDogLab/BitDogLab>. Acesso em: 23 set. 2024.
- [2] RASPBERRY PI FOUNDATION. RP2040 Datasheet. Disponível em: <https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf>. Acesso em: 23 set. 2024.
- [3] GAY, Warren. Debouncing. In: GAY, Warren. Exploring the Raspberry Pi 2 with C++. 1. ed. New York: Apress, 2015. p. 105-112.
- [4] IGINO, Wellington Passos et al. Ensino de sistemas embarcados baseado em projeto: exemplo aplicado à robótica. 2023.
- [5] CIRCUIT BASICS. How to set up a keypad on an Arduino. Disponível em: <https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/>. Acesso em: 25 set. 2024.

Exemplo 1

Controle de Led e Leitura de botão

Neste primeiro exemplo prático, vamos aprender como utilizar os GPIOs do Raspberry Pi Pico para controlar um LED e ler o estado de um botão. Essa atividade serve como uma introdução aos conceitos básicos de configuração e utilização dos pinos de entrada e saída, fundamentais para a interação com o mundo físico através de sistemas embarcados.

3.1.1. Objetivo do Exemplo

O objetivo deste exemplo é demonstrar como configurar um pino GPIO como saída para controlar um LED e outro pino como entrada para ler o estado de um botão. Através dessa prática, você será capaz de entender a lógica de funcionamento de um circuito simples e como programar o microcontrolador para reagir às interações do usuário.

Materiais Necessários

Para realizar este experimento, você precisará dos seguintes componentes:

- 1 x Raspberry Pi Pico W ,
- 1 x LED (preferencialmente de cor vermelha)
- 1 x Resistor de 330 ohms
- 1 x Botão de pressão (push button)
- 1 x Protoboard

Montagem do Circuito

A montagem do circuito deve ser feita conforme o diagrama apresentado na Figura 3.1.

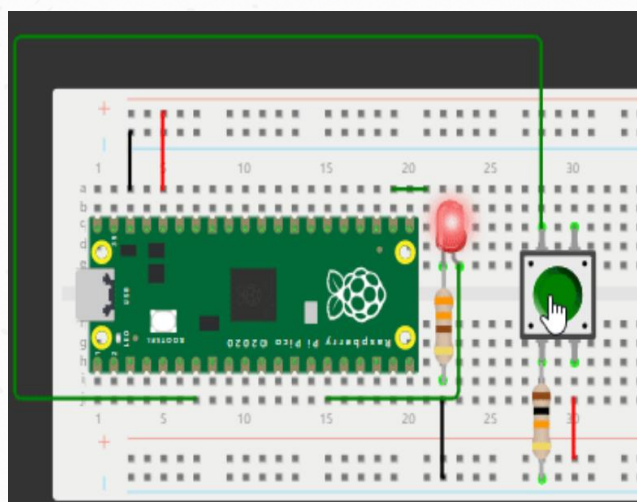


Figura 1.

A imagem mostra um circuito eletrônico montado em uma placa de prototipagem, também conhecida como "breadboard".

Imagine a placa como uma grade com vários furos, onde você pode inserir os componentes eletrônicos.

O circuito é composto por:

Um microcontrolador: um pequeno computador, representado por um chip verde com a logo da Raspberry Pi, localizado no centro da placa.

Um LED vermelho: uma lâmpada que emite luz vermelha, conectado à placa na coluna 25, próximo ao canto superior direito.

Um botão verde: um botão que pode ser pressionado, localizado à direita do LED, na coluna 30.

Um resistor: um componente que limita a corrente elétrica, representado por um retângulo marrom com uma banda dourada, conectado ao LED na coluna 21.

Um segundo resistor: outro resistor, também marrom com uma banda dourada, conectado ao botão na coluna 26.

Fios: representados por linhas verdes, conectando os componentes à placa e entre si.

Ao pressionar o botão, o circuito permite que a corrente flua para o LED, fazendo-o acender. O resistor limita a quantidade de corrente que passa pelo LED para evitar que ele queime.

O circuito demonstra um exemplo básico de como um botão pode controlar o funcionamento de um LED.

A imagem mostra uma representação gráfica do circuito, mas não fornece informações sobre as conexões específicas dos pinos do microcontrolador, que seriam necessárias para programar o seu comportamento.

	Componentes	GPIO
1	Led	GPIO11
2	Botões	GPIO5

Siga as instruções abaixo para montar o esquema eletrônico pela plataforma wokwi:

Conexão do LED:

- Conecte o terminal positivo(anodo)do LED ao pino GPIO 11 do Raspberry Pi Pico.

- O terminal negativo (catodo) deve ser conectado a um resistor de 330 ohms, e o outro terminal do resistor à linha de terra (GND) do Raspberry Pi Pico.

Conexão do Botão:

- Conecte uma perna do botão ao pino GPIO 5 do Raspberry Pi Pico.
- A outra perna do botão deve ser conectada ao GND.

Código do Exemplo

O código a seguir foi escrito em C e utiliza a biblioteca padrão do Raspberry Pi Pico para configurar os GPIOs e controlar o LED com base no estado do botão:

```
2 #include "pico/stdlib.h"
3
4 #define LED_PIN 11
5 #define BTN_PIN 5
6
7 int main()
8 {
9     // Inicializa o pino do LED e configura como saída
10    gpio_init(LED_PIN);
11    gpio_set_dir(LED_PIN, GPIO_OUT);
12
13    // Inicializa o pino do botão e configura como entrada
14    gpio_init(BTN_PIN);
15    gpio_set_dir(BTN_PIN, GPIO_IN);
16
17    // Loop infinito para verificar o estado do botão e controlar o LED
18    while (1)
19    {
20        while(gpio_get(BTN_PIN))
21        {
22            // Acende o LED se o botão estiver pressionado
23            gpio_put(LED_PIN, 1);
24        }
25        // Apaga o LED se o botão não estiver pressionado
26        gpio_put(LED_PIN, 0);
27    }
28 }
```

Figura 2. Fonte: imagem do autor

A imagem mostra um código em C para controlar um LED usando um botão, possivelmente em um microcontrolador Raspberry Pi Pico. Aqui está uma descrição detalhada dos elementos do código:

1. Inclusão de Biblioteca:

- A primeira linha (`#include "pico/stdlib.h"`) importa a biblioteca padrão da Raspberry Pi Pico para controlar as portas GPIO.

2. Definição de Pinos:

- As próximas linhas definem os pinos do LED e do botão:

- `LED_PIN` é o pino 11.

- `BTN_PIN` é o pino 5.

3. Função Principal:

- A função `main()` é o ponto de entrada do programa.

- Inicializa o pino do LED (`LED_PIN`) como saída e o pino do botão (`BTN_PIN`) como entrada usando as funções `gpio_init()` e `gpio_set_dir()`.

4. Loop Infinito:

- O código entra em um loop infinito (`while (1)`) que verifica continuamente o estado do botão.

- Dentro do loop, há outro loop (`while(gpio_get(BTN_PIN))`) que fica ativo enquanto o botão está pressionado. Nesse caso:

- O LED é aceso (`gpio_put(LED_PIN, 1)`).

- Quando o botão é liberado, o LED é apagado (`gpio_put(LED_PIN, 0)`).

Esse código é simples e eficiente para testar como ligar e desligar um LED com base na interação com um botão físico.

Análise do Código

Inclusão das Bibliotecas:

`#include "pico/stdlib.h"`: Inclui a biblioteca padrão para utilização das funções de controle dos GPIOs e temporização do Raspberry Pi Pico.

Definição dos Pinos:

`#define LED_PIN 11`: Define o pino 11 como o pino do LED.

`#define BTN_PIN 5`: Define o pino 5 como o pino do botão.

Inicialização dos Pinos:

`gpio_init()`: Inicializa o pino especificado.

`gpio_set_dir()`: Configura a direção do pino, se ele será utilizado como entrada (GPIO_IN) ou saída (GPIO_OUT).

Loop Principal:

`while (1)`: Cria um loop infinito para verificar continuamente o estado do botão.

`gpio_get(BTN_PIN)`: Lê o estado do pino do botão; retorna 1 se o botão estiver pressionado.

`gpio_put(LED_PIN, 1)`: Acende o LED quando o botão é pressionado.

`gpio_put(LED_PIN, 0)`: Apaga o LED quando o botão não é pressionado.

Explicação do Funcionamento

Neste exemplo, ao pressionar o botão, o estado lógico do pino GPIO 5 muda de 0 para 1, acionando o LED conectado ao pino GPIO 11. O resistor em série com o LED limita a corrente, protegendo o LED e o microcontrolador de possíveis danos. O código foi escrito de maneira a continuamente verificar o estado do botão, acendendo ou apagando o LED conforme a entrada detectada.

A partir deste ponto, você pode expandir o circuito e o código para incluir mais botões e LEDs, ou até mesmo sensores e atuadores adicionais, dependendo do seu projeto. No próximo subtópico, abordaremos o controle de LEDs e a leitura de dois botões, avançando para uma lógica de controle mais sofisticada.

Exemplo 2

Controle de Leds e Leitura de dois botões

Neste segundo exemplo prático, exploraremos como controlar vários leds utilizando dois botões como entradas. Esse experimento introduz conceitos mais avançados, como a combinação de sinais digitais para alterar as cores dos leds, ampliando as possibilidades de interação e controle em projetos com microcontroladores.

Objetivo do Exemplo

O objetivo desse exemplo é demonstrar como utilizar três pinos GPIO para controlar as cores de um LED RGB e dois pinos adicionais para ler o estado de dois botões. O sistema deve alterar a cor do LED RGB de acordo com os diferentes botões pressionados, permitindo a criação de uma interface visual interativa. Com esse exercício, você aprenderá a gerenciar múltiplos sinais de entrada e saída de maneira simultânea, utilizando o microcontrolador Raspberry Pi Pico.

Materiais Necessários

Para a realização desse experimento, serão necessários os seguintes componentes:

- 1x Raspberry Pi Pico
- 1x LED vermelho
- 1x LED verde
- 1x LED azul
- 3 x Resistores de 330 ohms (um para cada terminal de cor do LED)
- 2 x Botões de pressão (push button)

Montagem do Circuito

A montagem do circuito deve seguir o diagrama apresentado na Figura 3.2.

Diagrama de controle de LEDs RGB e leitura de dois botões.

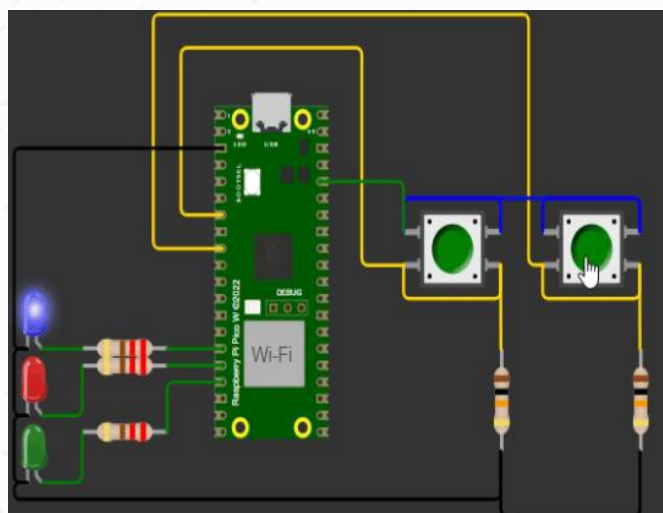


Figura 3. Fonte: imagem do autor

A imagem mostra um circuito eletrônico com um microcontrolador, LEDs e botões. Microcontrolador: um chip verde no centro da imagem, representando o "cérebro" do circuito.

LEDs: três LEDs coloridos (vermelho, azul e verde) conectados ao microcontrolador.

Botões: dois botões azuis, que provavelmente servem para controlar os LEDs.

Resistores: representados por retângulos marrons com uma banda dourada, conectados a cada LED.

Os fios amarelos conectam os componentes ao microcontrolador e entre si, formando o circuito. Os fios azuis provavelmente representam a conexão entre os botões e o microcontrolador.

O circuito demonstra a interação entre os componentes. Ao pressionar os botões, o microcontrolador provavelmente acende ou apaga os LEDs, dependendo da configuração do circuito.

A imagem não fornece informações sobre a programação do microcontrolador ou a função específica de cada botão.

Tabela 3. Fonte: Tabela elaborada pelo autor

	Componentes	GPIO
1	Led RGB	Vm - GPIO13 Az - GPIO12 Vd - GPIO11
2	Botões	GPIO05 e GPIO06

A tabela mostra a conexão entre os componentes de um dispositivo eletrônico e os pinos de entrada e saída do microcontrolador. Imagine que o microcontrolador é como o "cérebro" do dispositivo, e os pinos são como os seus "braços" e "mãos" que permitem que ele interaja com os outros componentes.

A primeira coluna da tabela lista o número do componente:

1: LED RGB: um LED que emite luz nas cores vermelha, verde e azul.

2: Botões: botões que podem ser pressionados para enviar sinais ao microcontrolador.

A segunda coluna da tabela lista o nome do componente:

LED RGB: um LED que emite luz nas cores vermelha, verde e azul.

Botões: botões que podem ser pressionados para enviar sinais ao microcontrolador.

A terceira coluna da tabela lista o pino GPIO específico que é usado para controlar cada componente.

GPIO: é como um terminal, uma entrada ou saída, do microcontrolador.

Vm - GPIO13, Az - GPIO12, Vd - GPIO11: os pinos que controlam o LED RGB, O "Vm" provavelmente representa a cor vermelha, "Az" a azul e "Vd" a verde.

GPIO05 e GPIO06: os pinos que controlam os botões.

A tabela é como um mapa, mostrando a conexão entre os componentes e o microcontrolador. Ela ajuda a entender como os diferentes elementos do dispositivo são interligados e controlados pelo microcontrolador.

Os leds vermelhos (Vm), azul (az) e verde(vd) serão conectados a três pinos GPIO do Raspberry Pi Os dois botões serão conectados a outros dois pinos GPIO para detectar as interações do usuário.

Conexão dos LED vermelho, verde e azul:

- Conecte o terminal positivo (anodo) do LED vermelho ao pino GPIO 13 através de um resistor de 330 ohms, e o terminal negativo (catodo) ao GND.
- Conecte o terminal positivo do LED verde ao pino GPIO 11 através de um resistor de 330 ohms, e o terminal negativo ao GND.
- Conecte o terminal positivo do LED azul ao pino GPIO 12 através de um resistor de 330 ohms, e o terminal negativo ao GND.
- Conecte o terminal azul (B) ao pino GPIO 13 através de um resistor de 330 ohms.

Conexão dos Botões:

- Conecte uma perna do primeiro botão ao pino GPIO 5 e a outra perna ao GND.
- Conecte uma perna do segundo botão ao pino GPIO 6 e a outra perna ao GND.

Código do Exemplo

O código abaixo foi escrito em C Bare metal e utiliza a biblioteca padrão do Raspberry Pi Pico. Ele configura os GPIOs para controlar os três LEDs e ler o estado dos dois botões. Com base no estado dos botões, o código acende diferentes combinações de LEDs.

Figura - Código de Exemplo em C.

Figura 5. Fonte: Imagem do autor

A imagem contém um código em C, usado para controlar três LEDs (vermelho, verde e azul) e dois botões. Aqui está a descrição detalhada:

Inclusão de Biblioteca:

- A linha "#include "pico/stdlib.h"" importa a biblioteca padrão da Raspberry Pi Pico para controle de GPIO.

Definição de Pinos:

- As seguintes linhas definem os pinos utilizados:

- 'LED_R_PIN 12': Pino do LED vermelho.

- 'LED_G_PIN 13': Pino do LED verde.

- 'LED_B_PIN 11': Pino do LED azul.

- 'BTN_A_PIN 5': Pino do botão A.

- 'BTN_B_PIN 6': Pino do botão B.

Função 'set_leds':

- A função 'set_leds' aceita três parâmetros booleanos para ligar/desligar os LEDs vermelho, verde e azul.

- 'gpio_put(LED_R_PIN, red)': Controla o LED vermelho.

- 'gpio_put(LED_G_PIN, green)': Controla o LED verde.

- 'gpio_put(LED_B_PIN, blue)': Controla o LED azul.

Função Principal 'main()':

- Inicializa os pinos dos LEDs como saída com 'gpio_init()' e 'gpio_set_dir()'.

- Inicializa os pinos dos botões como entrada, com configuração de "pull-down" (resistores internos) para garantir que os botões estejam desligados quando não pressionados.

- 'gpio_pull_down(BTN_A_PIN)' e 'gpio_pull_down(BTN_B_PIN)'.

Este código é parte de um programa que controla três LEDs e dois botões em uma placa Raspberry Pi Pico, permitindo configurar cada LED com base na lógica aplicada aos botões.

```

2 #include "pico/stdlib.h"
3
4 #define LED_R_PIN 12
5 #define LED_G_PIN 13
6 #define LED_B_PIN 11
7 #define BTN_A_PIN 5
8 #define BTN_B_PIN 6
9
10 void set_leds(bool red, bool green, bool blue) {
11     gpio_put(LED_R_PIN, red);
12     gpio_put(LED_G_PIN, green);
13     gpio_put(LED_B_PIN, blue);
14 }
15
16 int main() {
17     // Inicializa os pinos dos LEDs como saída
18     gpio_init(LED_R_PIN);
19     gpio_set_dir(LED_R_PIN, GPIO_OUT);
20     gpio_init(LED_G_PIN);
21     gpio_set_dir(LED_G_PIN, GPIO_OUT);
22     gpio_init(LED_B_PIN);
23     gpio_set_dir(LED_B_PIN, GPIO_OUT);
24
25     // Inicializa os pinos dos botões como entrada
26     gpio_init(BTN_A_PIN);
27     gpio_set_dir(BTN_A_PIN, GPIO_IN);
28     gpio_pull_down(BTN_A_PIN);
29     gpio_init(BTN_B_PIN);
30     gpio_set_dir(BTN_B_PIN, GPIO_IN);
31     gpio_pull_down(BTN_B_PIN);
  
```


Figura 6. Fonte: Imagem do autor

A imagem mostra uma parte de um código em C que controla o estado de LEDs com base no estado de dois botões. Abaixo está a descrição detalhada:

Loop Principal

O código contém um loop 'while (true)' que verifica continuamente o estado dos dois botões e aciona os LEDs de acordo.

1. Verificação de ambos os botões:

- Se os dois botões (A e B) estiverem pressionados ao mesmo tempo:

```

// C
if (gpio_get(BTN_A_PIN) && gpio_get(BTN_B_PIN)) {
    set_leds(1, 1, 1); // Todos os LEDs acesos (branco)
}

```

- Todos os LEDs (vermelho, verde e azul) são acesos, resultando na cor branca.

2. Verificação do botão A:

- Se apenas o botão A estiver pressionado:

```

// C
else if (gpio_get(BTN_A_PIN)) {
    set_leds(1, 0, 0); // Apenas o LED vermelho aceso
}

```

- Somente o LED vermelho é aceso.

3. Verificação do botão B:

- Se apenas o botão B estiver pressionado:

```

// C
else if (gpio_get(BTN_B_PIN)) {
    set_leds(0, 1, 0); // Apenas o LED verde aceso
}

```

- Somente o LED verde é aceso.

4. **Nenhum botão pressionado:**

- Se nenhum dos botões estiver pressionado:

```

// C
else {
    set_leds(0, 0, 1); // Apenas o LED azul aceso
}

```

- Somente o LED azul é aceso.

Pausa

- Após cada verificação e alteração do estado dos LEDs, o código aguarda 100 milissegundos:

```

// C
sleep_ms(100);

```

Este código usa lógica condicional para controlar os LEDs com base no estado de dois botões, permitindo diferentes combinações de cores ao pressionar os botões.

```

33 // Loop principal para verificar o estado dos botões e controlar os LEDs
34 while (true) {
35     if (gpio_get(BTN_A_PIN) && gpio_get(BTN_B_PIN)) {
36         set_leds(1, 1, 1); // Todos os LEDs acesos (branco)
37     } else if (gpio_get(BTN_A_PIN)) {
38         set_leds(1, 0, 0); // LED vermelho aceso
39     } else if (gpio_get(BTN_B_PIN)) {
40         set_leds(0, 1, 0); // LED verde aceso
41     } else {
42         set_leds(0, 0, 1); // LED azul aceso
43     }
44     sleep_ms(100);
45 }
46 return 0;
47 }

```

Análise do Código

Inclusão das Bibliotecas:

- **#include "pico/stdlib.h":** Inclui a biblioteca padrão do Raspberry Pi Pico, que contém as funções necessárias para configurar e controlar os GPIOs.

Definição dos Pinos:

- **#define LED_R_PIN 11, LED_G_PIN 12, LED_B_PIN 13:** Define os pinos para os LEDs vermelho, verde e azul, respectivamente.
- **#define BTN_A_PIN 5, BTN_B_PIN 6:** Define os pinos para os botões de controle.

Função set_leds():

- **Recebe** três parâmetros booleanos (red, green e blue) e configura os pinos correspondentes para ligar ou desligar cada LED.

Inicialização dos Pinos:

- `gpio_init()` e `gpio_set_dir()`: Inicializam os pinos dos LEDs como saída e os dos botões como entrada.
- `gpio_pull_down()`: Configura os pinos dos botões com resistores pull-down, garantindo que o estado padrão dos pinos seja zero quando os botões não estiverem pressionados.

Loop Principal:

- Verifica continuamente o estado dos botões e altera o estado dos LEDs conforme as condições:
- Ambos os botões pressionados: Todos os LEDs acendem, formando a cor branca.
- Apenas o botão A pressionado: Somente o LED vermelho acende.
- Apenas o botão B pressionado: Somente o LED verde acende.
- Nenhum botão pressionado: Somente o LED azul acende.

Explicação do Funcionamento

Neste exemplo, os três LEDs de diferentes cores acendem em diferentes combinações conforme os estados dos botões. Cada cor é controlada individualmente pelos pinos GPIO, e o sistema combina os LEDs acesos com base nos sinais de entrada recebidos dos botões. Quando nenhum botão é pressionado, apenas o LED azul acende.

Ao pressionar o botão A, acende-se o LED vermelho; ao pressionar o botão B, acende-se o LED verde; e ao pressionar ambos os botões simultaneamente, todos os LEDs acendem, emitindo luz branca. Este exemplo demonstra como múltiplos sinais de entrada e saída podem ser gerenciados simultaneamente, permitindo a criação de interfaces mais interativas e dinâmicas.

A partir deste ponto, você pode experimentar diferentes combinações de LEDs e adicionar mais botões para ampliar as opções de controle. No próximo subtópico, exploraremos a leitura de um teclado matricial, que permitirá capturar comandos mais complexos utilizando um número reduzido de pinos GPIO, viabilizando a construção de interfaces de usuário ainda mais sofisticadas.

Exemplo 3

Leitura de Teclado Matricial

Neste terceiro exemplo prático, exploraremos como ler um teclado matricial 4x4 utilizando o Raspberry Pi Pico. Esse exemplo demonstra como capturar entradas mais complexas com um número reduzido de pinos GPIO, viabilizando a criação de interfaces de usuário interativas e eficientes para sistemas embarcados.

Objetivo do Exemplo

O objetivo deste exemplo é demonstrar como utilizar oito pinos GPIO do Raspberry Pi Pico para capturar o estado de 16 teclas dispostas em um teclado matricial 4x4. Você aprenderá a mapear cada tecla pressionada para um caractere correspondente, utilizando técnicas de varredura (scanning) de linhas e colunas. Com esse exercício, você será capaz de desenvolver sistemas que utilizam teclados matriciais para entrada de dados, criando interfaces de usuário mais sofisticadas e interativas.

Materiais Necessários

Para a realização deste experimento, serão necessários os seguintes componentes:

- 1 x Raspberry Pi Pico
- 1 x Teclado matricial 4x4 (16 teclas)

Montagem do Circuito

A montagem do circuito deve seguir o diagrama apresentado na Figura 3.3 e Figura 3.4. O teclado matricial possui 8 pinos: 4 correspondentes às colunas e 4 às linhas. Esses pinos serão conectados aos GPIOs do Raspberry Pi Pico para realizar a varredura e identificar quais teclas estão sendo pressionadas.

1. Conexão das Linhas do Teclado:

- Conecte os pinos das linhas (R1, R2, R3, R4) aos pinos GPIO 8, 7, 6 e 5 do Raspberry Pi Pico, respectivamente.

2. Conexão das Colunas do Teclado:

- Conecte os pinos das colunas (C1, C2, C3, C4) aos pinos GPIO 4, 3, 2 e 1 do Raspberry Pi Pico, respectivamente.

Teclado matricial 4x4[5]

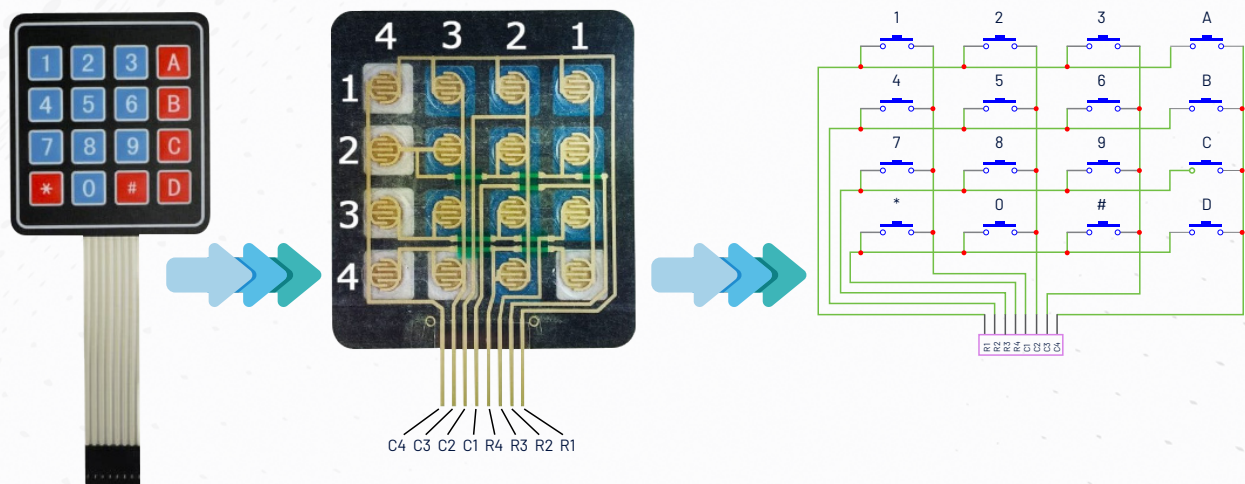


Figura 7. Diagrama de leitura de teclado matricial 4x4 + rasp rp2040. Fonte: imagem do autor

Esta é uma imagem que mostra os diagramas de um teclado de membrana de 4x4. A imagem da esquerda mostra um teclado de membrana de 4x4 típico. A imagem do meio mostra os componentes internos do teclado de membrana. Na imagem da direita, os componentes internos do teclado de membrana são organizados em um diagrama para mostrar o caminho dos sinais através do teclado. O diagrama mostra 4 linhas, rotuladas como C1 a C4, e 4 colunas, rotuladas como R1 a R4. Cada botão no teclado de membrana está conectado a um único cruzamento de uma linha e uma coluna. Por exemplo, o botão "1" está conectado à linha C4 e à coluna R1. O diagrama mostra que quando um botão é pressionado, uma conexão é feita entre a linha e a coluna correspondentes, criando um caminho para o sinal passar.

O teclado de membrana possui 16 botões organizados em uma grade de 4 por 4. Os botões são rotulados com números de 1 a 9, bem como os símbolos de asterisco, zero, cerquilha e D. Cada botão é destacado com uma cor diferente: vermelho, azul, azul escuro ou azul claro.

As conexões de linha e coluna são mostradas em verde no diagrama. Os botões são conectados a um conjunto de 8 pinos que são mostrados como um retângulo vermelho na parte inferior do diagrama. O conjunto de pinos é rotulado com os números de 1 a 8.

A imagem fornece uma representação visual de como um teclado de membrana funciona e como os botões são conectados a linhas e colunas.

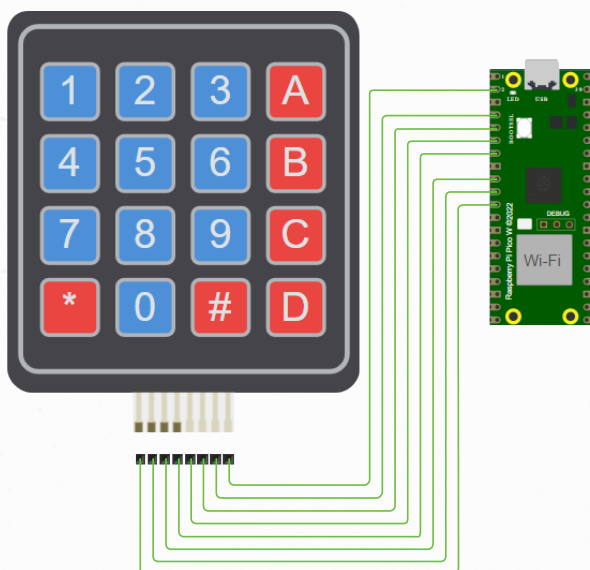


Figura 8. Fonte: imagem do autor

A imagem mostra um teclado de membrana de 4x4 conectado a um Raspberry Pi Pico W. O teclado tem 16 botões, organizados em uma grade de 4x4, rotulados com os números de 1 a 9, bem como os símbolos de asterisco, zero, cerquilha e D. Cada botão é destacado com uma cor diferente: vermelho, azul, azul escuro ou azul claro.

O Raspberry Pi Pico W é um pequeno computador de placa única (SBC) que pode ser usado para uma variedade de projetos. Ele possui um microcontrolador ARM, Wi-Fi integrado e uma variedade de pinos de E/S.

O teclado está conectado ao Raspberry Pi Pico W por meio de um conjunto de fios verdes. Os fios estão conectados aos pinos GPIO do Raspberry Pi Pico W e aos pinos do teclado.

	Teclado Matricial	Rasp. Pi Pico
1	C4	GP1
2	C3	GP2
3	C2	GP3
4	C1	GP4
5	R4	GP5
6	R3	GP6
7	R2	GP7
8	R1	GP8

Tabela 4. Fonte: Tabela elaborada pelo autor

A tabela mostra a conexão entre um teclado matricial e o microcontrolador Raspberry Pi Pico. Imagine o teclado matricial como um conjunto de botões organizados em linhas e colunas. O microcontrolador é como o "cérebro" do dispositivo que identifica as teclas pressionadas. A primeira coluna da tabela representa a ordem das conexões.

A segunda coluna, "Teclado Matricial", descreve as linhas e colunas do teclado, usando as letras "C" para coluna e "R" para linha. Cada célula da tabela representa uma tecla do teclado.

A terceira coluna, "Rasp. Pi Pico", indica os pinos GPIO (General Purpose Input/Output) do Raspberry Pi Pico que são usados para se comunicar com o teclado.

Por exemplo, na primeira linha:

"C4" significa que o pino do teclado é da coluna 4.

"GP1" indica que o pino GPIO 1 do Raspberry Pi Pico está conectado a essa coluna do teclado.

A tabela funciona como um mapa, mostrando a conexão específica entre cada tecla do teclado e o pino correspondente no Raspberry Pi Pico. É por meio dessas conexões que o microcontrolador detecta quais teclas são pressionadas, permitindo que ele interprete os comandos e informações do teclado.

Código do Exemplo

O código abaixo foi escrito em C e utiliza a biblioteca padrão do Raspberry Pi Pico. Ele configura os GPIOs para escanear as linhas e colunas do teclado matricial e identificar qual tecla está sendo pressionada.

Figura - Código de exemplo em C

```

2  #include "pico/stdlib.h"
3
4  // Definição dos pinos do teclado matricial
5  #define ROWS 4
6  #define COLS 4
7
8  // Atualização dos pinos de acordo com as novas conexões
9  const uint8_t row_pins[ROWS] = {8, 7, 6, 5}; // Pinos conectados às linhas R1, R2, R3, R4
10 const uint8_t col_pins[COLS] = {4, 3, 2, 1}; // Pinos conectados às colunas C1, C2, C3, C4
11
12 // Mapa de teclas do teclado matricial
13 const char key_map[ROWS][COLS] = {
14     {'1', '2', '3', 'A'},
15     {'4', '5', '6', 'B'},
16     {'7', '8', '9', 'C'},
17     {'*', '0', '#', 'D'}
18 };
19
20 // Função para inicializar os pinos do teclado
21 void keypad_init() {
22     for (int i = 0; i < ROWS; i++) {
23         gpio_init(row_pins[i]);
24         gpio_set_dir(row_pins[i], GPIO_OUT);
25         gpio_put(row_pins[i], 0); // Configura as linhas como saídas em nível baixo
26     }
27     for (int i = 0; i < COLS; i++) {
28         gpio_init(col_pins[i]);
29         gpio_set_dir(col_pins[i], GPIO_IN);
30         gpio_pull_down(col_pins[i]); // Configura as colunas como entradas com pull-down
31     }
32 }

```

Figura 9. Fonte: imagem do autor

A imagem mostra um trecho de código em C, provavelmente para um microcontrolador. O código define como um teclado matricial é configurado e usado. No início, o código inclui uma biblioteca chamada "pico/stdlib.h", que provavelmente contém funções para o microcontrolador. Em seguida, são definidas as constantes "ROWS" e "COLS" com o valor 4, indicando que o teclado tem 4 linhas e 4 colunas. Depois, são definidas duas variáveis "row_pins" e "col_pins" do tipo "uint8_t", que provavelmente armazenam os números dos pinos do microcontrolador conectados às linhas e colunas do teclado, respectivamente. Os valores dentro dos colchetes indicam os pinos específicos, como 8, 7, 6 e 5 para as linhas e 4, 3, 2 e 1 para as colunas. Um mapa de teclas chamado "key_map" é definido, que provavelmente relaciona cada combinação de linha e coluna do teclado a um caractere. Os caracteres estão entre aspas simples, como '1', '2', '3', 'A', etc. A função "keypad_init" é definida para inicializar os pinos do teclado. O código dentro da função configura cada pino das linhas como uma saída com nível baixo e cada pino das colunas como uma entrada com pull-down. Em resumo, o código define a configuração do teclado matricial, incluindo os pinos do microcontrolador que são usados para controlá-lo, e cria um mapa de teclas para relacionar cada tecla do teclado a um caractere. O código está escrito em linguagem de programação C, que é comumente usada em programação de microcontroladores.



```

34 // Função para ler o teclado matricial
35 char read_keypad() {
36     for (int row = 0; row < ROWS; row++) {
37         gpio_put(row_pins[row], 1); // Ativa a linha atual
38         for (int col = 0; col < COLS; col++) {
39             if (gpio_get(col_pins[col])) { // Verifica se a coluna atual está em nível alto
40                 gpio_put(row_pins[row], 0); // Desativa a linha atual
41                 return key_map[row][col]; // Retorna a tecla pressionada
42             }
43         }
44         gpio_put(row_pins[row], 0); // Desativa a linha atual
45     }
46     return '\0'; // Retorna nulo se nenhuma tecla foi pressionada
47 }

```

Figura 10. Fonte: imagem do autor

A imagem mostra um trecho de código escrito em C, que implementa uma função para ler um teclado matricial. A função se chama 'read_keypad()' e tem o seguinte comportamento:

1. Início da função: É declarada como 'char', o que significa que retornará um caractere que representa a tecla pressionada no teclado matricial.
 2. Laço para percorrer as linhas: Um loop 'for' percorre as linhas do teclado. O loop vai de 0 até o número total de linhas (denotado por 'ROWS').
- Em cada iteração, a linha atual é ativada usando a função 'gpio_put(row_pins[row], 1)'.
 3. Laço para percorrer as colunas: Outro loop 'for' aninhado percorre as colunas do teclado. O loop vai de 0 até o número total de colunas (denotado por 'COLS').
- Para cada coluna, o código verifica se a tecla correspondente está pressionada, usando 'gpio_get(col_pins[col])'.
 4. Verificação da tecla pressionada: Se a coluna estiver ativa (tecla pressionada), o código:
- Desativa a linha atual ('gpio_put(row_pins[row], 0)').
- Retorna o valor correspondente à tecla pressionada, que está mapeado na variável 'key_map[row][col]'.
 5. Final do loop: Após a verificação das colunas, o código desativa a linha atual.
 6. Retorno padrão: Caso nenhuma tecla tenha sido pressionada, a função retorna '\0', indicando que nada foi detectado.
- Essa função basicamente percorre todas as linhas e colunas do teclado matricial e identifica se alguma tecla foi pressionada.

```

49 int main() {
50     stdio_init_all();
51     keypad_init(); // Inicializa o teclado matricial
52
53     while (true) {
54         char key = read_keypad(); // Lê o teclado
55         if (key != '\0') { // Verifica se uma tecla foi pressionada
56             printf("Tecla pressionada: %c\n", key); // Imprime a tecla pressionada
57             sleep_ms(200); // Aguarda para evitar múltiplas leituras da mesma tecla
58         }
59     }
60     return 0;
61 }

```

Figura 11. Fonte: imagem do autor

A imagem apresenta um trecho de código em C que implementa a função 'main()' para operar um teclado matricial. A seguir, está a descrição das principais partes do código:

1. Início da função: A função 'main()' começa com a declaração 'int main()', que é o ponto de entrada do programa.
 2. Inicialização:
- 'stdio_init_all()': Inicializa as funções de entrada e saída padrão.
- 'keypad_init()': Chama uma função para inicializar o teclado matricial.
 3. Loop infinito: O programa entra em um loop 'while (true)', que continuará executando indefinidamente.
 4. Leitura do teclado: Dentro do loop, o código chama a função 'read_keypad()', que lê a entrada do teclado e armazena o resultado na variável 'key'.
 5. Verificação da tecla pressionada:
- O código verifica se a tecla pressionada é diferente de '\0' (o que indica que nenhuma tecla foi pressionada). Isso é feito na linha 'if (key != '\0')'.
 6. Impressão da tecla pressionada:
- Se uma tecla foi pressionada, o programa imprime a tecla na tela com a linha 'printf("Tecla pressionada: %c\n", key)'.
 7. Delay para evitar leituras múltiplas: O código então aguarda 200 milissegundos usando 'sleep_ms(200);' para evitar a detecção múltipla da mesma tecla.
 8. Finalização: O código termina com 'return 0;', que indica que o programa foi encerrado com sucesso, embora, no caso de um loop infinito, essa linha não seja alcançada.
- Essencialmente, este trecho de código configura um loop que continuamente lê o estado do teclado matricial e imprime a tecla pressionada sempre que uma nova tecla é detectada.

Análise do Código

1. Inclusão das Bibliotecas:

- '#include "pico/stdlib.h"': Inclui a biblioteca padrão do Raspberry Pi Pico, que contém as funções necessárias para configurar e controlar os GPIOs e para realizar operações de entrada e saída.

2. Definição dos Pinos e do Mapa de Teclas:

- `const uint8_t row_pins[ROWS]` e `col_pins[COLS]`: Define os pinos utilizados para as linhas e colunas do teclado matricial.
- `const char key_map[ROWS][COLS]`: Mapa de teclas que associa cada combinação de linha e coluna a um caractere específico.

3. Função `keypad_init()`:

- Inicializa os pinos das linhas como saídas e os das colunas como entradas, configurando as colunas com resistores de pull-down para garantir que o estado padrão seja zero.

4. Função `read_keypad()`:

- Realiza a varredura do teclado ativando cada linha uma de cada vez e verificando se alguma coluna está em nível alto. Se uma tecla for pressionada, retorna o caractere correspondente do mapa de teclas.

5. Loop Principal:

- Lê continuamente o estado do teclado. Se uma tecla for pressionada, imprime o caractere correspondente na saída padrão (terminal) e aguarda 200 milissegundos para evitar múltiplas leituras consecutivas da mesma tecla.

Explicação do Funcionamento

Neste exemplo, o Raspberry Pi Pico realiza uma varredura das linhas e colunas do teclado matricial para identificar qual tecla está sendo pressionada. A técnica de varredura funciona ativando uma linha por vez e verificando quais colunas estão em nível alto. Cada tecla do teclado matricial está posicionada na interseção de uma linha e uma coluna, permitindo identificar precisamente qual tecla foi pressionada. O uso do mapa de teclas (`key_map`) permite traduzir a posição de cada tecla no teclado para o caractere correspondente. Quando uma tecla é detectada, seu caractere é exibido no terminal, facilitando o desenvolvimento de interfaces interativas e sistemas de controle mais complexos.

/Importante/

A partir deste ponto, você pode expandir o código para detectar combinações de teclas, implementar senhas ou mesmo criar uma calculadora.

/Importante/



UPDF
WWW.UPDF.COM



embarca
tech | Residência
Tecnológica
em Sistemas
Embarcados

