embarca tech | Residência Tecnológica em Sistemas Embarcados

Práticas de Portas de Entrada e Saída - GPIO

Unidade 4 | Capítulo 2

Executores:
INSTITUTO FEDERAL Piauí
INSTITUTO FEDERAL Rio Grande do Norte
INSTITUTO FEDERAL Maranhão
INSTITUTO FEDERAL Ceará
cepedi
INSTITUTO HARDWARE BR

Coordenação: Softex

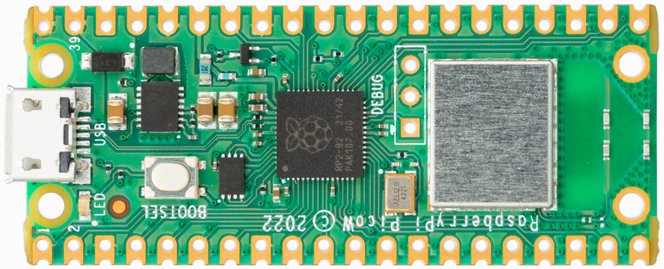
Iniciativa: MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

Sumário

Revisão:
Plataforma BitDogLab

BitDogLab

- Componentes
  - » Bateria
  - » Display OLED 128x64 pixels
  - » Matriz de LEDs coloridos
  - » Microfone
  - » Joystick
  - » Botões
  - » Buzzer
  - » Conectores de expansão



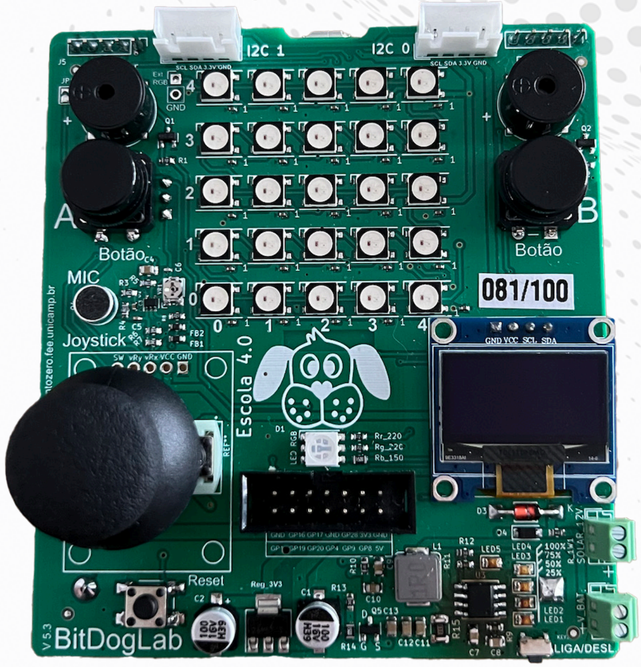


Fig. 1. Plataforma BitDogLab

A imagem mostra uma placa de desenvolvimento Raspberry Pi Pico W, .
Componentes Principais:
No canto superior direito, há uma porta micro-USB, utilizada para alimentação e transferência de dados.
Perto da porta USB, encontra-se um botão marcado como "BOOTSEL", utilizado para colocar o dispositivo em modo de carregamento de firmware.
No centro da placa, há um chip microcontrolador RP2040 com o logotipo do Raspberry Pi.
À esquerda do chip, há um módulo de metal, que abriga o chip Wi-Fi, uma das principais diferenças da versão "W" da placa.
Vários outros componentes menores, como capacitores e resistores, estão distribuídos pela placa.
Pinos de Conexão:
A placa tem pinos de conexão (GPIO) em ambas as bordas, em um total de 20 pinos de cada lado. Esses pinos são utilizados para conectar a placa a outros componentes e sensores externos.
Marcas:
Na parte superior, lê-se "Raspberry Pi Pico W © 2022", indicando que é a versão com conectividade sem fio lançada em 20

Fig. 2 - BitDogLab

A imagem mostra uma placa eletrônica verde com diversos componentes. Na parte superior da placa, há uma matriz de 4x4 botões pequenos, totalizando 16 botões. À esquerda da matriz, há um joystick, semelhante ao de controles de videogame. Logo abaixo do joystick, há alguns conectores e componentes menores.
Na parte direita da placa, ao lado da matriz de botões, há uma pequena tela (provavelmente uma tela OLED) e dois botões maiores acima dela. Próximo à parte inferior da placa, há uma série de conectores e componentes eletrônicos adicionais, incluindo um conector central.
A placa tem as marcas "BitDogLab" na parte inferior, e há outros textos e logotipos técnicos impressos na superfície.

Fonte: https://github.com/BitDogLab/BitDogLab/raw/main/doc/illustrations/joystick-assembled.png

GPIOs - General Purpose Input/Outputs

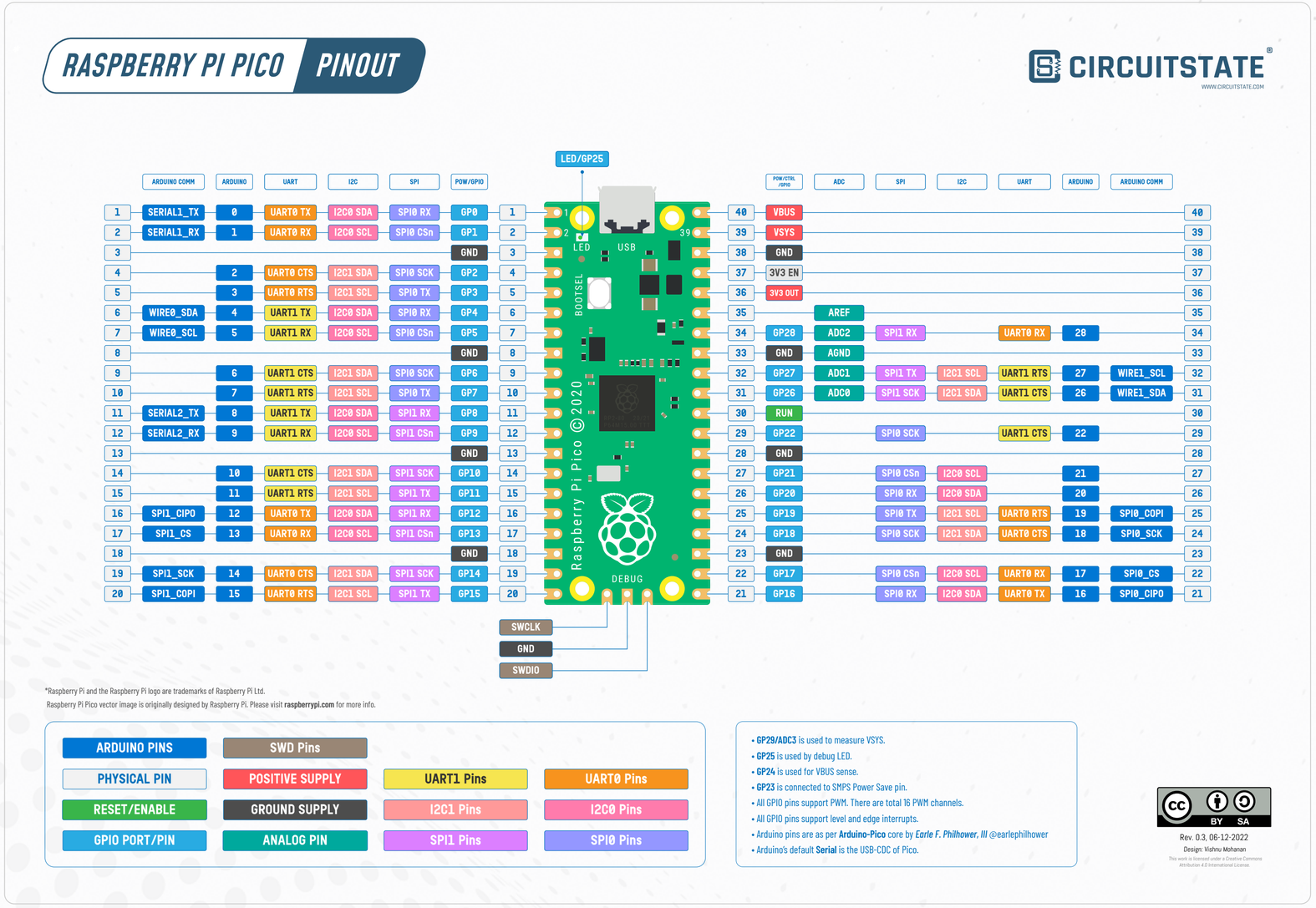


Fig. 3. GPIOs - Entradas/Saídas de Uso Geral

Fonte: Raspberry-Pi-Pico-Pinout-r0.2 (circuitstate.com)

A imagem apresenta um diagrama esquemático de um microcontrolador, especificamente o Raspberry Pi Pico. É uma representação visual de um circuito eletrônico, mostrando a disposição física dos pinos e suas respectivas funções.
Elementos principais:
Pinos: Linhas finas que se estendem dos lados da placa. Cada pino representa um ponto de conexão para componentes externos.
Rótulos: Textos curtos que identificam cada pino. Os rótulos mais comuns são:
GPIO: Pinos de uso geral, que podem ser configurados como entrada ou saída.
GND: Pinos de terra, utilizados como referência de voltagem zero.
VCC: Pinos de alimentação, que fornecem tensão para o circuito.
UART: Pinos para comunicação serial, utilizados para enviar e receber dados.
SWD: Pinos para depuração, utilizados para programar e testar o microcontrolador.
Cores: As cores são usadas para diferenciar os diferentes tipos de pinos. Por exemplo, os pinos GPIO podem ser de uma cor, os pinos de alimentação de outra, etc.
Legenda: Uma pequena caixa de texto na parte inferior da imagem que explica a simbologia utilizada no diagrama.
Descrição detalhada:
A imagem mostra um retângulo que representa a placa do microcontrolador. Ao longo das bordas do retângulo, há linhas finas que se estendem para fora, representando os pinos. Cada pino tem um rótulo que indica sua função. A maioria dos pinos são GPIOs, que são os pinos mais versáteis do microcontrolador. Os pinos GPIO podem ser configurados para receber sinais de entrada (como um botão) ou para enviar sinais de saída (como acender um LED). Além dos GPIOs, a imagem mostra também pinos de alimentação, pinos de terra e pinos para comunicação serial.

GPIOs - General Purpose Input/Outputs

Características:

- O RP2040 tem 30 pinos de entrada / saída de uso geral (GPIO);
- Conversores - ADC / DAC;
- Controladores de Interrupção;
- Embora seja possível configurar as propriedades dos pinos via software, alguns pinos já vêm configurados por padrão;
- Além disso, alguns pinos possuem características específicas que os tornam mais ou menos adequados para diferentes projetos.

GPIOs - General Purpose Input/Outputs

A escolha do pino dependerá das necessidades especificas de um projeto. Segue abaixo uma guia de GPIO:

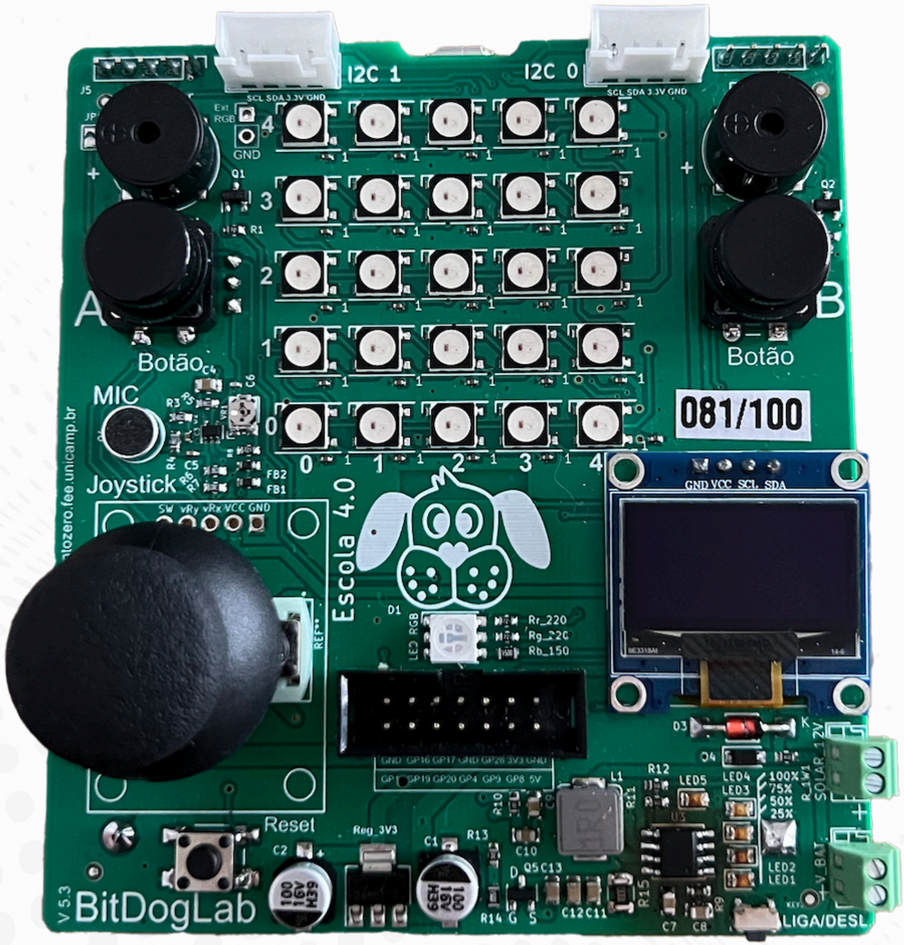
Tipo de Pino	Função	Exemplos de Uso
GPIO	Entrada/Saída geral	Controlar LEDs, ler botões, comunicar com sensores
Serial (UART)	Comunicação serial	Conectar-se a módulos Bluetooth, GPS, etc.
I2C	Comunicação serial de multi-master	Conectar-se a displays OLED, sensores de temperatura, etc.
SPI	Comunicação serial síncrona	Conectar-se a displays LCD, sensores de toque, etc.
PWM	Modulação por largura de pulso	Controlar a velocidade de motores DC, ajustar o brilho de LEDs, etc.

Fig. 4. Tipos de pinos e suas funções

Esta tabela descreve os diferentes tipos de pinos encontrados em microcontroladores e suas funções e usos. GPIO é o tipo mais básico de pino e pode ser configurado como entrada ou saída para controlar LEDs, ler botões, comunicar com sensores. Serial (UART) permite comunicação serial com outros dispositivos como módulos Bluetooth, GPS, etc. I2C é um protocolo de comunicação serial multi-master, ideal para conectar displays OLED, sensores de temperatura, etc. SPI é outro protocolo de comunicação serial, mas síncrona, usado para conectar displays LCD, sensores de toque, etc. PWM usa modulação por largura de pulso para controlar a velocidade de motores DC, ajustar o brilho de LEDs, etc

RaspberryPi Pico

Explorando os GPIOscom BitDoglab;



	Componentes	GPIO
1	Leds RGB	Vm -GPIO13 Az -GPIO12 Vd -GPIO11
2	Comunicação SPI	Tx -GP19 Rx -GPIO16 CSn -GPIO17 SCK -GP18
3	Display OLED	A -GPIO14 SCL -GPIO15
4	Botões	GPIO5 e GPIO6
5	Buzzer	GPIO10 eGPIO21
6	Matriz de Leds Coloridos	GPIO7 , 5 linhas por 5 colunas
7	Microfone	GP28
8	Joystick	Vry -GPIO26 Vrx -GPIO27 Sw -GPIO22

Fig. 5. BitDoglab.

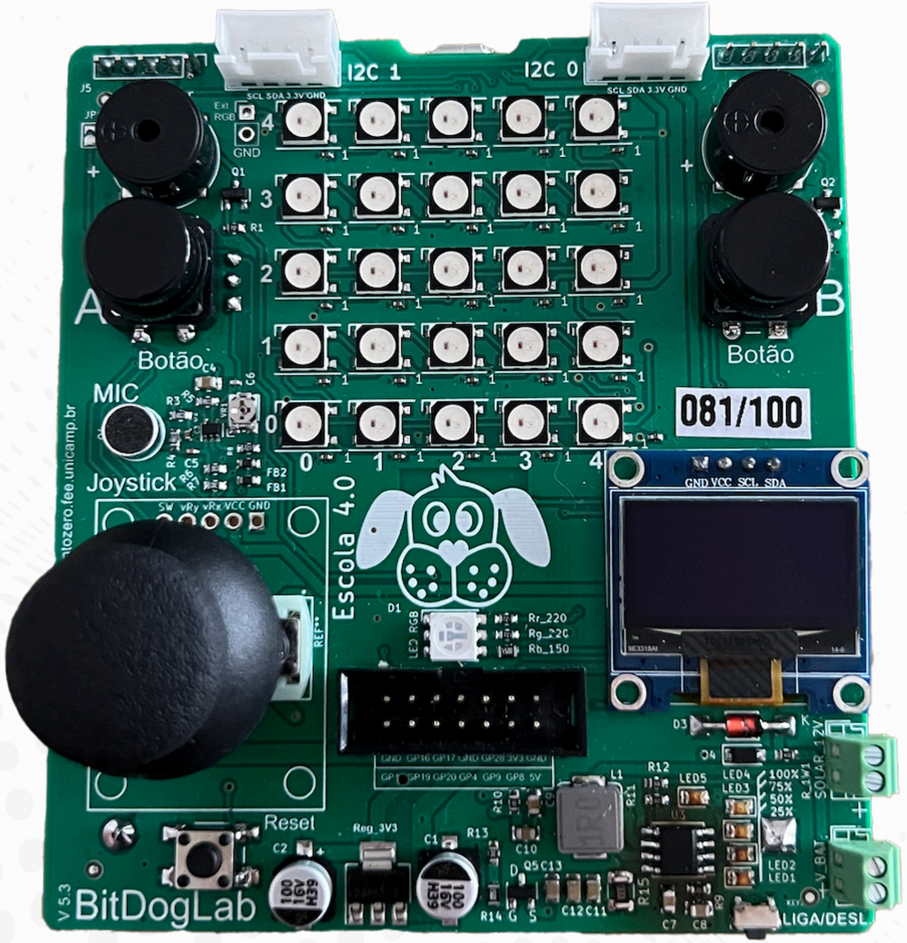
A imagem mostra uma placa eletrônica verde com diversos componentes. Na parte superior da placa, há uma matriz de 4x4 botões pequenos, totalizando 16 botões. À esquerda da matriz, há um joystick, semelhante ao de controles de videogame. Logo abaixo do joystick, há alguns conectores e componentes menores.
Na parte direita da placa, ao lado da matriz de botões, há uma pequena tela (provavelmente uma tela OLED) e dois botões maiores acima dela. Próximo à parte inferior da placa, há uma série de conectores e componentes eletrônicos adicionais, incluindo um conector central.
A placa tem as marcas "BitDogLab" na parte inferior, e há outros textos e logotipos técnicos impressos na superfície.

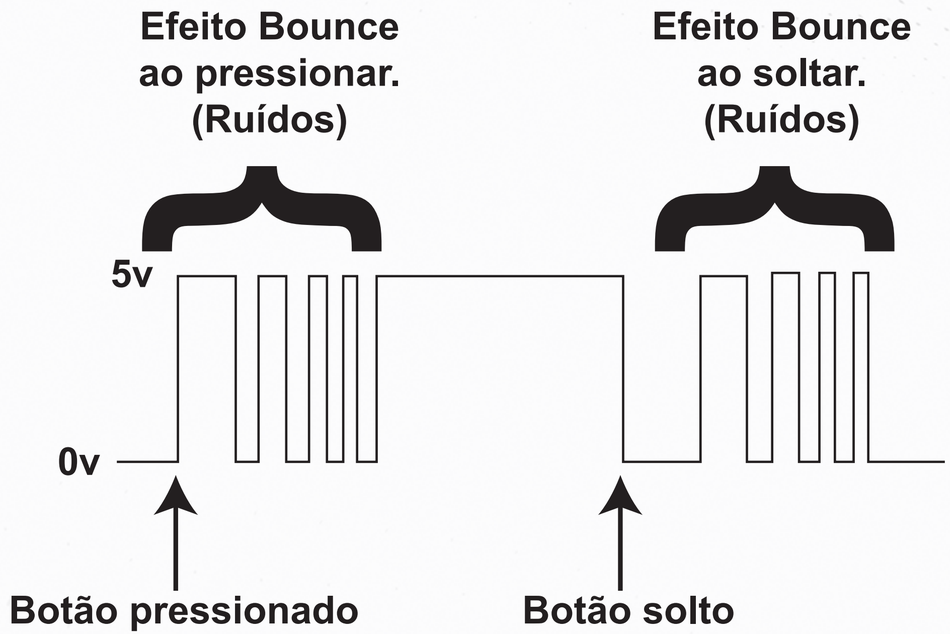
Fig. 6. GPIO e seus respectivos componentes.

Esta tabela mostra quais pinos GPIO em um microcontrolador são usados para controlar vários componentes.
Componente GPIO
Leds RGB Vm - GPIO13, Az - GPIO12, Vd - GPIO11
Comunicação SPI Tx - GP19, Rx - GPIO16, CSn - GPIO17, SCK - GP18
Display OLED SDA - GPIO14, SCL - GPIO15
Botões GPIO5 e GPIO6
Buzzer GPIO10 e GPIO21
Matriz de LEDs Coloridos GPIO7, 5 linhas por 5 colunas
Microfone GP28
Joystick Vry - GPIO26, Vrx - GPIO27, Sw - GPIO22

Efeito que ocorre ao pressionar o botão 1 vez

Importante entender antes de fazer as práticas





Fonte: https://www.blogdarobotica.com/2021/05/24/entenda-o-efeito-bounce-e-como-fazer-debounce-no-arduino-ao-apertar-um-botao/

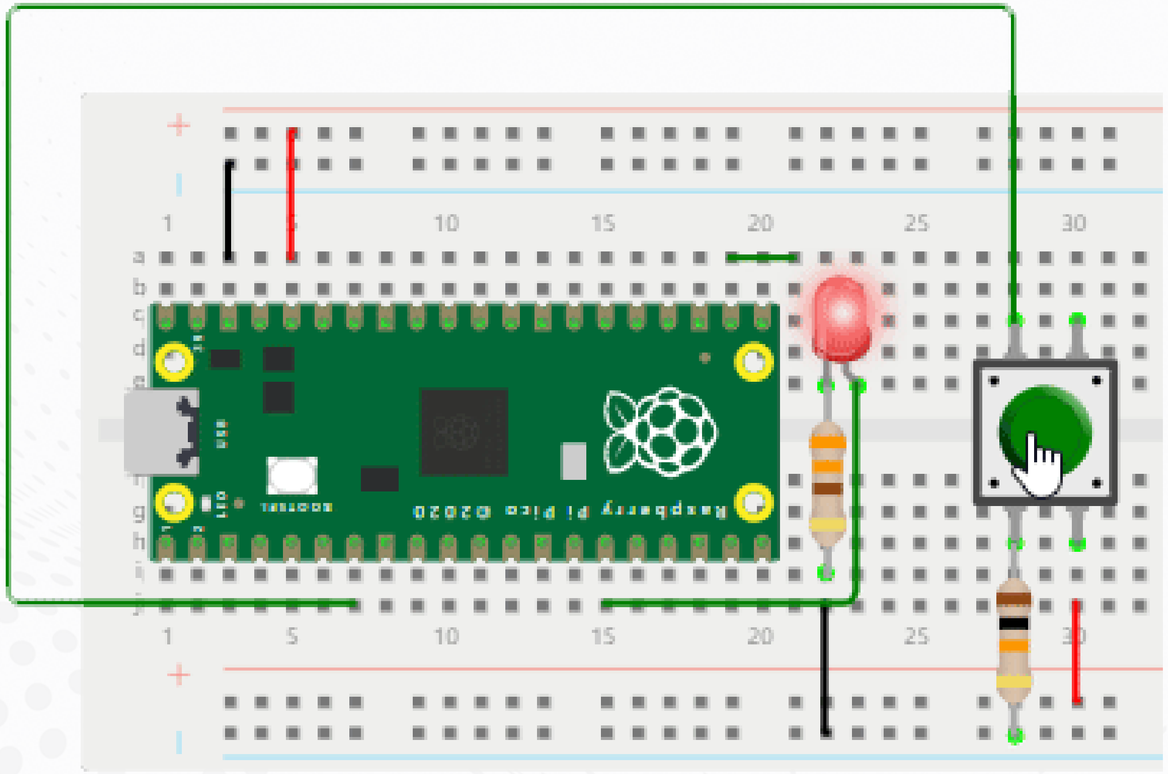
Fig. 7. BitDoglab e o efeito Bounce.

A imagem está dividida em duas partes.
Parte 1 (à esquerda):
Mostra uma placa eletrônica com vários componentes:
- Na parte superior, há uma matriz de **botões dispostos em linhas e colunas**, com dois botões maiores posicionados em cima à esquerda e à direita.
- Há um **joystick** no canto inferior esquerdo da placa.
- Também estão presentes outros componentes eletrônicos como microfone, portas I2C, e entradas/saias digitais, entre outros.

Parte 2 (à direita):
Contém uma explicação gráfica sobre o **efeito "bounce" ao pressionar e soltar um botão**. O efeito "bounce" se refere aos pequenos ruídos ou oscilações elétricas que ocorrem quando o botão é pressionado ou solto, antes de estabilizar no estado "alto" (5V) ou "baixo" (0V).
- O gráfico mostra uma linha horizontal que representa o estado de 0V (botão não pressionado). Quando o botão é pressionado, há oscilações (ou ruídos) antes da linha se estabilizar em 5V. O mesmo efeito ocorre ao soltar o botão, com oscilações que aparecem antes de voltar ao estado 0V.
Abaixo dessa explicação está escrito: ***Importante entender antes de fazer as práticas***, ressaltando que o fenômeno "bounce" deve ser compreendido para evitar problemas ao trabalhar com botões em projetos eletrônicos.
Em resumo, a imagem combina uma representação física de uma placa eletrônica com botões e joystick, junto com uma explicação teórica sobre o comportamento elétrico ao pressionar e soltar um botão.

Exercícios práticos:

Exemplo 1: Controle de Led e leitura de botão;



Fonte: https://wokwi.com/pi-pico

```c
#include "pico/stdlib.h"

#define LED_PIN 11
#define BTN_PIN 5

int main()
{
    gpio_init(LED_PIN);
    gpio_set_dir(LED_PIN, GPIO_OUT);
    gpio_init(BTN_PIN);
    gpio_set_dir(BTN_PIN, GPIO_IN);
    while (1)
    {
        while(gpio_get(BTN_PIN))
        {
            gpio_put(LED_PIN, 1);
        }
        gpio_put(LED_PIN, 0);
    }
}
```

Fig. 8. Raspberry Pi Pico e controle de LED.
Fonte: https://wokwi.com/pi-pico

A imagem mostra um diagrama de um circuito simples.
O circuito contém os seguintes componentes:
Um Raspberry Pi Pico, que é um microcontrolador.
Um botão, que é um dispositivo de entrada.
Um LED vermelho, que é um dispositivo de saída.
Um LED verde, que é um dispositivo de saída.
Três resistores, que são dispositivos passivos que restringem o fluxo de corrente.
O botão está conectado ao Raspberry Pi Pico, que controla os LEDs. O LED vermelho está ligado a um resistor que está conectado ao Raspberry Pi Pico. O LED verde está ligado a um resistor que está conectado ao Raspberry Pi Pico e a um resistor que está conectado ao botão.
O circuito está montado em uma placa de prototipagem, que permite que você conecte componentes de maneira fácil e rápida.
O diagrama também mostra o layout do circuito, que pode ser usado para construir o circuito em uma placa de prototipagem.
O diagrama mostra que os LEDs vermelho e verde estão ligados a diferentes pinos do Raspberry Pi Pico. Isso significa que os LEDs podem ser controlados independentemente um do outro.
O diagrama também mostra que o botão é conectado a um pino que está conectado ao LED verde. Isso significa que o LED verde acenderá quando o botão for pressionado.

Fig. 9. Código em Linguagem C p/ acender o LED.
Fonte: imagem do autor

Esta imagem mostra um código em linguagem C para um microcontrolador, provavelmente um Raspberry Pi Pico. O código define um circuito simples com um LED e um botão, e faz o LED acender quando o botão é pressionado.
Vamos analisar o código passo a passo:
Linhas 1 e 2: Inclui a biblioteca padrão do Raspberry Pi Pico, "pico/stdlib.h".
Linhas 3 e 4: Define os pinos GPIO do microcontrolador para o LED e o botão. O LED está no pino 11 e o botão no pino 5.
Linha 6: Define a função principal do programa, chamada "main()".
Linhas 8-11: Inicializa os pinos GPIO definidos anteriormente. O LED é configurado como saída ("GPIO_OUT") e o botão como entrada ("GPIO_IN").
Linhas 12-19: Entra em um loop infinito (while(1)) para executar o código repetidamente.
Linhas 13-17: Entra em outro loop (while) que verifica constantemente o estado do botão. Se o botão estiver pressionado (gpio_get(BTN_PIN) retorna verdadeiro), o LED é ligado (gpio_put(LED_PIN, 1)).
Linha 18: Se o botão não estiver pressionado, o LED é desligado (gpio_put(LED_PIN, 0)).
Em resumo, o código configura o microcontrolador para ler o estado do botão e acender o LED quando o botão estiver pressionado.

Exercícios práticos:

Exemplo 2: Controle de Leds RGB e leiturade dois botões;

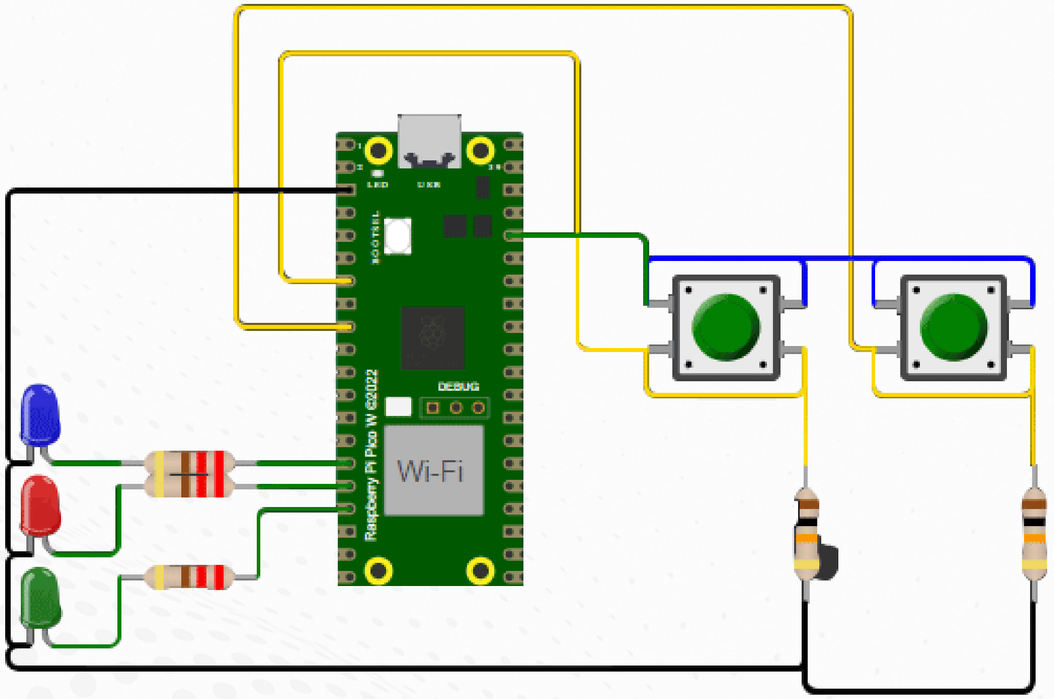


Fig. 10. Controle de Leds RGB e leitura de dois botões
Fonte: https://wokwi.com/projects/409242558191197185

Esta é uma imagem de um circuito que foi desenhado em um fundo cinza escuro. Na parte superior direita da imagem, há um chip verde com "Raspberry Pi Pico W GP22" escrito nele. Há três LEDs conectados ao chip em uma linha vertical no lado esquerdo do chip. O LED superior é azul, o LED do meio é vermelho e o LED inferior é verde. Cada LED está conectado ao chip com um fio verde e a um resistor marrom, vermelho, marrom, com um fio verde. Há dois botões cinza, que cada um possui um botão verde dentro, localizados à direita do chip. Cada botão está conectado ao chip com um fio amarelo, um fio azul e um fio verde. Eles também são conectados a um resistor marrom, vermelho, marrom com um fio preto. O fio amarelo do botão superior está conectado ao chip no pino 22 e o fio amarelo do botão inferior está conectado ao chip no pino 19. O fio azul de ambos os botões está conectado a um fio azul que está conectado ao pino 20 do chip. O fio verde de cada botão está conectado a um fio verde que está conectado ao pino 21 do chip. O chip tem um fio amarelo que sai do pino 18 do chip. O fio se curva para cima e para a direita e, em seguida, para baixo e para a esquerda e se conecta ao pino 16 do chip. Um fio amarelo adicional sai do pino 12 do chip. O fio se curva para cima e para a direita, e então para baixo e para a esquerda, e, em seguida, para cima e para a direita novamente e se conecta ao pino 17 do chip. Finalmente, um fio amarelo adicional sai do pino 14 do chip e se curva para cima e para a direita.

```c
#include <stdio.h>
#include <pico/stdlib.h>

// Pinos dos componentes
#define BUTTON_A_PIN 5
#define BUTTON_B_PIN 6
#define RED_LED_PIN 12
#define GREEN_LED_PIN 13
#define BLUE_LED_PIN 11
```

Fig. 11. Código em Linguagem C p/ Controle de Leds RGB.
Fonte: imagem do autor

A imagem mostra um trecho de código escrito em C, onde estão sendo definidas algumas configurações para um microcontrolador, provavelmente o Raspberry Pi Pico. Aqui está uma descrição das linhas presentes:
1. `include <stdio.h>`:
 - Inclui a biblioteca padrão de entrada e saída em C, que permite usar funções como `printf` para imprimir textos no console.
2. `#include <pico/stdlib.h>`:
 - Inclui a biblioteca padrão do Raspberry Pi Pico, que contém funções úteis para manipulação de hardware, como os pinos GPIO.
3. Comentários (// Pinos dos componentes):
 - Indica que o trecho a seguir define os pinos usados para os componentes eletrônicos conectados.
4.Definição dos pinos com `#define`:
 - `BUTTON_A_PIN 5`: Define o pino 5 como o pino de conexão para o botão A.
 - `BUTTON_B_PIN 6`: Define o pino 6 como o pino de conexão para o botão B.
 - `RED_LED_PIN 12`: Define o pino 12 como o pino de conexão para o LED vermelho.
 - `GREEN_LED_PIN 13`: Define o pino 13 como o pino de conexão para o LED verde.
 - `BLUE_LED_PIN 11`: Define o pino 11 como o pino de conexão para o LED azul.
Esse código inicializa os pinos GPIO que serão usados para controlar dois botões e três LEDs (vermelho, verde e azul) conectados ao microcontrolador.

Exercícios práticos:

Exemplo 2: Controle de Leds RGB e leiturade dois botões;

```c
int main() {
    // Inicializa os LEDs como saídas
    gpio_init(LED_PIN_VM);
    gpio_set_dir(LED_PIN_VM, GPIO_OUT);

    gpio_init(LED_PIN_VD);
    gpio_set_dir(LED_PIN_VD, GPIO_OUT);

    gpio_init(LED_PIN_AZ);
    gpio_set_dir(LED_PIN_AZ, GPIO_OUT);

    // Inicializa os botões como entradas com pull-down
    gpio_init(BUTTON_PIN_A);
    gpio_set_dir(BUTTON_PIN_A, GPIO_IN);
    gpio_pull_down(BUTTON_PIN_A);

    gpio_init(BUTTON_PIN_B);
    gpio_set_dir(BUTTON_PIN_B, GPIO_IN);
    gpio_pull_down(BUTTON_PIN_B);

    while (true) {
        // Verifica se o botão A foi pressionado
        if (gpio_get(BUTTON_PIN_A)) {
            // Botão A pressionado, acende o LED azul
            gpio_put(LED_PIN_AZ, 1);
            gpio_put(LED_PIN_VM, 0);
            gpio_put(LED_PIN_VD, 0);
            sleep_ms(100); // Atraso para evitar múltiplas leituras
        } else if (gpio_get(BUTTON_PIN_B)) {
            // Botão B pressionado, acende o LED verde
            gpio_put(LED_PIN_AZ, 0);
            gpio_put(LED_PIN_VM, 1);
            gpio_put(LED_PIN_VD, 0);
            sleep_ms(100); // Atraso para evitar múltiplas leituras
        } else {
            // Nenhum botão pressionado, apaga todos os LEDs
            gpio_put(LED_PIN_AZ, 0);
            gpio_put(LED_PIN_VM, 0);
            gpio_put(LED_PIN_VD, 0);
        }
    }
}
```

Fig. 12: Código em Linguagem C p/ Controle de Leds RGB.
Fonte: https://wokwi.com/projects/409242558191197185

A imagem contém duas seções de código C, aparentemente para controle de LEDs e botões usando GPIO. Vou descrever o que aparece de forma geral:
À esquerda (função main): Inicialização de LEDs como saídas:
Três LEDs (LED_PIN_VM, LED_PIN_VD, LED_PIN_AZ) são configurados para saída com gpio_init() e gpio_set_dir().
Inicialização de botões como entradas com pull-down:
Dois botões (BUTTON_PIN_A, BUTTON_PIN_B) são configurados para entrada com gpio_init() e gpio_set_dir(), e o resistor de pull-down é ativado com gpio_pull_down().
À direita (loop while): Verificação de botões:
Se o botão A for pressionado (gpio_get(BUTTON_PIN_A)), o LED azul (LED_PIN_AZ) é ligado, enquanto os outros são apagados. Há um pequeno atraso de 100 ms.
Se o botão B for pressionado (gpio_get(BUTTON_PIN_B)), o LED verde (LED_PIN_VD) é ligado, com o mesmo comportamento de apagar os outros LEDs e um atraso de 100 ms.
Se nenhum dos botões for pressionado, todos os LEDs são apagados. O código parece controlar LEDs com base na entrada de botões.

Exercícios práticos:

Exemplo 3: Leitura de teclado matricial.

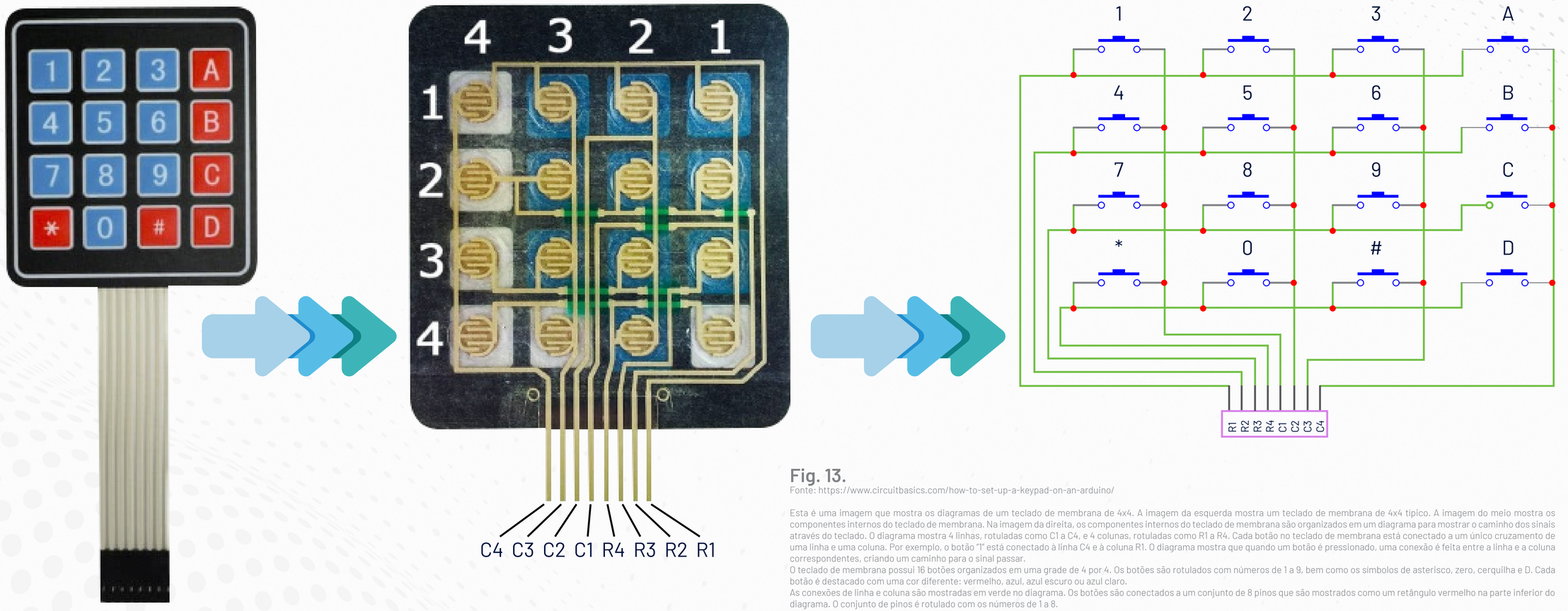


Fig. 13.
Fonte: https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/

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.

Exercícios práticos:

Exemplo 3: Leitura de teclado matricial.

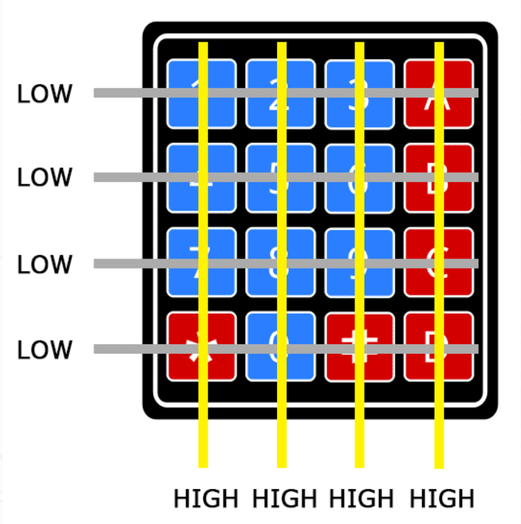


Fig. 14

A imagem mostra um diagrama de um teclado de membrana de 4x4. O teclado tem 16 botões organizados em uma grade de 4x4, rotulados com os números 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 diagrama mostra as linhas do teclado em amarelo, com a linha C1 na parte superior e a linha C4 na parte inferior. As colunas do teclado são mostradas em cinza, com a coluna R1 à esquerda e a coluna R4 à direita. O diagrama mostra as linhas configuradas como LOW e as colunas configuradas como HIGH. Em baixo da imagem, diz "Passo 1".

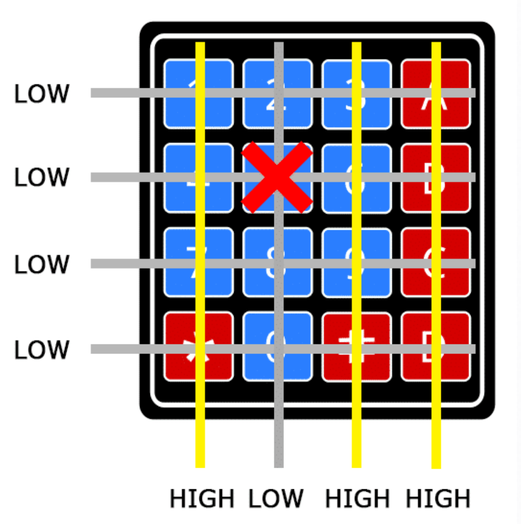


Fig. 15

A imagem mostra um diagrama de um teclado de membrana de 4x4. O teclado tem 16 botões organizados em uma grade de 4x4, rotulados com os números 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 botão "5" está marcado com um X vermelho. O diagrama mostra as linhas do teclado em amarelo, com a linha C1 na parte superior e a linha C4 na parte inferior. As colunas do teclado são mostradas em cinza, com a coluna R1 à esquerda e a coluna R4 à direita. A linha C2 está configurada como LOW e as colunas R2, R3 e R4 estão configuradas como HIGH. As linhas C1, C3 e C4 e a coluna R1 estão configuradas como HIGH. Em baixo da imagem, diz "Passo 2".

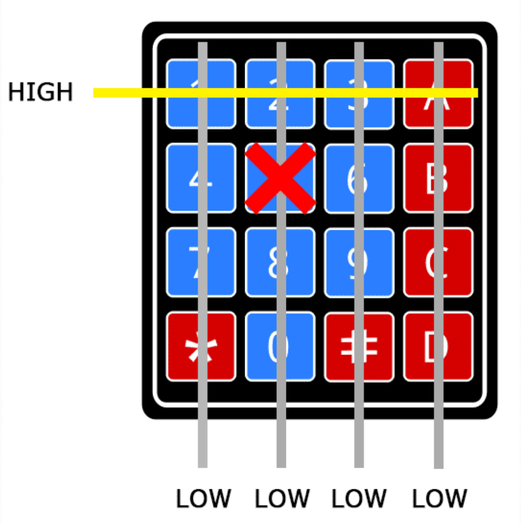


Fig. 16

A imagem mostra um diagrama de um teclado de membrana de 4x4. O teclado tem 16 botões organizados em uma grade de 4x4, rotulados com os números 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 botão "5" está marcado com um X vermelho. O diagrama mostra as linhas do teclado em amarelo, com a linha C1 na parte superior e a linha C4 na parte inferior. As colunas do teclado são mostradas em cinza, com a coluna R1 à esquerda e a coluna R4 à direita. A linha C1 está configurada como HIGH e as colunas R1, R2, R3 e R4 estão configuradas como LOW. As linhas C2, C3 e C4 estão configuradas como LOW. Em baixo da imagem, diz "Passo 3".

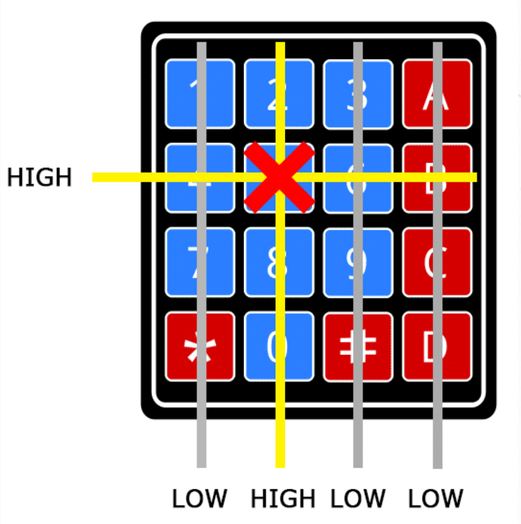


Fig. 17

A imagem mostra um diagrama de um teclado de membrana de 4x4. O teclado tem 16 botões organizados em uma grade de 4x4, rotulados com os números 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 botão "5" está marcado com um X vermelho. O diagrama mostra as linhas do teclado em amarelo, com a linha C1 na parte superior e a linha C4 na parte inferior. As colunas do teclado são mostradas em cinza, com a coluna R1 à esquerda e a coluna R4 à direita. A linha C1 está configurada como HIGH, a coluna R2 está configurada como HIGH, e as colunas R1, R3 e R4 estão configuradas como LOW. As linhas C2, C3 e C4 estão configuradas como LOW. Em baixo da imagem, diz "Passo 4".

Fonte: https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/

Exercícios práticos:

Exemplo 3: Leitura de teclado matricial.

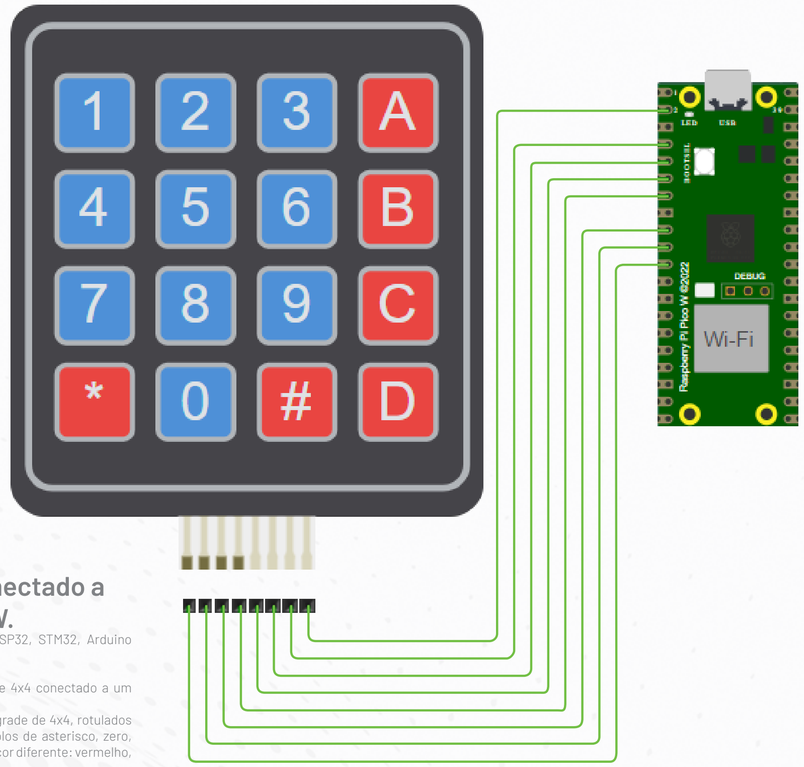


Fig. 18. Teclado 4x4 conectado a um Raspberry Pi Pico W.
Fonte: rp2040 - teclado matricial - Wokwi ESP32, STM32, Arduino Simulator

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	CA	GP1
2	C3	GP2
3	C2	GP3
4	C1	GP4
5	R4	GP5
6	R3	GP6
7	R2	GP7
8	R1	GP8

Fig. 19. como conectar um teclado matricial a um Raspberry Pi Pico
Fonte: imagem do autor

A tabela mostra como conectar um teclado matricial a um Raspberry Pi Pico. Ela tem três colunas:
A primeira coluna é numerada de 1 a 8, representando os pinos de conexão do teclado.
A segunda coluna mostra os pinos do teclado, que são organizados em linhas (C1 a C4) e colunas (R1 a R4).
A terceira coluna mostra os pinos correspondentes no Raspberry Pi Pico, chamados de GP1 a GP8.
Cada linha da tabela mostra a conexão entre um pino específico do teclado e um pino específico do Raspberry Pi Pico. Por exemplo, a linha 1 mostra que o pino 1 do teclado (C4) deve ser conectado ao pino GP1 do Raspberry Pi Pico.

Exercícios práticos:

Exemplo 3: Leitura de teclado matricial.

Fig. 20.
Fonte: imagem do autor

```c
#include <stdio.h>
#include <pico/stdlib.h>

// Define os pinos das colunas do teclado
const uint8_t COL_PINS[] = {1, 2, 3, 4};
const uint8_t ROW_PINS[] = {5, 6, 7, 8};

// Mapa de teclas
const char KEY_MAP[4][4] = {
    {'D', '#', '0', '*'},
    {'C', '9', '8', '7'},
    {'B', '6', '5', '4'},
    {'A', '3', '2', '1'}
};
```

A imagem mostra um código escrito em linguagem de programação C. O código define um mapa de teclas para um teclado de membrana de 4x4.
O código começa com duas linhas que incluem arquivos de cabeçalho:
#include <stdio.h>: este arquivo inclui a biblioteca de entrada/saída padrão, que fornece funções para interagir com dispositivos de entrada e saída.
#include <pico/stdlib.h>: este arquivo inclui a biblioteca padrão do Raspberry Pi Pico, que fornece funções para interagir com o hardware do Pico.
O código então define duas matrizes:
COL_PINS: esta matriz contém os números dos pinos que estão conectados às colunas do teclado. Os pinos são definidos como {1, 2, 3, 4}.
ROW_PINS: esta matriz contém os números dos pinos que estão conectados às linhas do teclado. Os pinos são definidos como {5, 6, 7, 8}.
O código então define uma matriz bidimensional chamada KEY_MAP. Esta matriz contém um mapa de teclas para o teclado, com cada célula da matriz representando um botão no teclado. A matriz é definida como:
const char KEY_MAP[4][4] = {
{'D', '#', '0', '*'},
{'C', '9', '8', '7'},
{'B', '6', '5', '4'},
{'A', '3', '2', '1'}
};
content_copy Use code
https://support.google.com/legal/answer/13505487.C

O código define a matriz KEY_MAP com as seguintes teclas:
Linha 1: 'D', '#', '0', '*'
Linha 2: 'C', '9', '8', '7'
Linha 3: 'B', '6', '5', '4'
Linha 4: 'A', '3', '2', '1'

Este código pode ser usado para ler os dados do teclado de membrana e identificar qual tecla está sendo pressionada. Quando um botão é pressionado, o código pode ser usado para determinar a linha e a coluna do botão pressionado e, em seguida, usar a matriz KEY_MAP para identificar qual tecla foi pressionada.

```c
// Função para ler uma tecla do teclado
char read_keypad(uint8_t *cols, uint8_t *rows) {
  for (int i = 0; i < 4; i++) {
    gpio_put(rows[i], 0); // Define o pino da linha como baixo
    uint8_t result = 0;
    for (int j = 0; j < 4; j++) {
      result |= gpio_get(cols[j]); // Lê os estados dos pinos das colunas
    }
    if (result == 0) { // Verifica se alguma coluna está baixa
      // Obtém a tecla usando a posição do bit menos significativo
      char key = KEY_MAP[i][__builtin_ctz(result)];
      gpio_put(rows[i], 1); // Define o pino da linha como alto (debounce)
      return key;
    }
    gpio_put(rows[i], 1); // Define o pino da linha como alto
  }
  return 0; // Nenhuma tecla pressionada
}
```

Fig. 21.
Fonte: imagem do autor.

A imagem mostra um código escrito em linguagem C, que define uma função chamada read_keypad para ler uma tecla de um teclado matricial conectado a um microcontrolador.
A função recebe dois parâmetros:
cols: um ponteiro para uma matriz de inteiros que representa os pinos conectados às colunas do teclado.
rows: um ponteiro para uma matriz de inteiros que representa os pinos conectados às linhas do teclado.
A função funciona da seguinte maneira:
Configura as linhas do teclado como LOW: o código itera sobre os pinos das linhas (definidos na matriz rows) e configura cada um deles como LOW usando a função gpio_put. Isso significa que os pinos estão em estado de baixa tensão, permitindo que a corrente flua através deles para as colunas do teclado.
Lê os estados dos pinos das colunas: O código itera sobre os pinos das colunas (definidos na matriz cols) e lê o estado de cada um deles usando a função gpio_get. Esta função retorna um valor binário para cada pino, indicando se ele está alto (HIGH) ou baixo (LOW).
Verifica se alguma coluna está baixa: O código verifica se o valor retornado pela função gpio_get é igual a 0. Se for, isso significa que pelo menos uma coluna está em estado LOW.
Obtém a tecla pressionada: O código usa a função builtin_ctz para encontrar a posição do bit menos significativo (LSB) que está em 0 no valor retornado pela função gpio_get. Esta posição corresponde à coluna do teclado que está em estado LOW. O código então usa a matriz KEY_MAP e a posição da coluna para determinar qual tecla foi pressionada.
Configura a linha como HIGH (debounce): Depois de identificar a tecla, o código configura o pino da linha correspondente como HIGH usando a função gpio_put. Esta etapa serve para eliminar o efeito de "rebotamento" (debounce) que pode ocorrer quando um botão é pressionado e liberado rapidamente.
Retorna a tecla: A função retorna o caractere correspondente à tecla pressionada.
Se nenhuma tecla for pressionada, a função retorna 0.

- Código completo: https://wokwi.com/projects/409296514249449473

Link único

O link fornecedo leva a um projeto no Wokwi, uma plataforma online para simulação de circuitos eletrônicos. O projeto em questão é um circuito simples que demonstra o funcionamento de um motor DC controlado por um botão.

Principais Aspectos

- GPIOs e sua importância
- Plataforma BitDogLab
- Exercícios práticos
- Projetos possíveis