embarca tech | Residência Tecnológica em Sistemas Embarcados

Introdução ao Desenvolvimento de Software Embarcado com Microcontroladores

Unidade 4 | Capítulo 1

Otávio Alcântara de Lima Júnior

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

Unidade 4

Capítulo 1

1. BOAS-VINDAS E INTRODUÇÃO

Olá, estudantes!

Boas-vindas ao material de apoio sobre Microcontroladores, criado para complementar e enriquecer o conteúdo do Capítulo 1 da Unidade 4.

Os objetivos da nossa conversa hoje são:

- Entender as características do desenvolvimento de software em microcontroladores.
- Conhecer e caracterizar a placa de desenvolvimento BitDogLab e seus componentes principais, incluindo a CPU, memória e interfaces de entrada/saída.
- Compreender como as bibliotecas de desenvolvimento do Raspberry Pi Pico estão organizadas.

Microcontroladores são dispositivos digitais altamente versáteis, amplamente utilizados no desenvolvimento de sistemas embarcados.

Essencialmente, um microcontrolador integra em um único chip um sistema de computação compacto, que inclui um ou mais processadores, memória de programa, memória de dados e diversos periféricos dedicados a tarefas de controle e comunicação.

Por serem simples, esses dispositivos têm baixo custo e consumo de energia, tornando-os ideais para diversas aplicações, desde eletrodomésticos até sistemas industriais complexos. Por isso, os microcontroladores são a escolha preferida para soluções econômicas e eficientes.
Outra vantagem significativa dos microcontroladores é a **simplicidade de sua programação.** Atualmente, os fornecedores de semicondutores

oferecem muitas bibliotecas e kits de desenvolvimento de software (SDKs) que abstraem a complexidade dos circuitos internos, proporcionando aos desenvolvedores interfaces amigáveis e de fácil utilização.

Essas bibliotecas não só simplificam o processo de desenvolvimento, mas também podem ser integradas com pilhas de protocolos de comunicação e sistemas operacionais de tempo real (RTOS).

Isso permite a criação de aplicações mais complexas, que atendem a requisitos rigorosos de tempo real e comunicação, expandindo as possibilidades de uso dos microcontroladores em projetos sofisticados e desafiadores.

Embora os microcontroladores ofereçam várias vantagens, é importante estarmos cientes dos desafios inerentes ao seu desenvolvimento. **Uma das principais dificuldades reside nas limitações computacionais e de memória desses dispositivos.**

Ao contrário dos computadores de propósito geral, os microcontroladores possuem poder de **processamento** e capacidade de **memória restritos**, o que impõe ao projetista de sistemas embarcados a necessidade de otimizar o código de forma cuidadosa e eficiente.

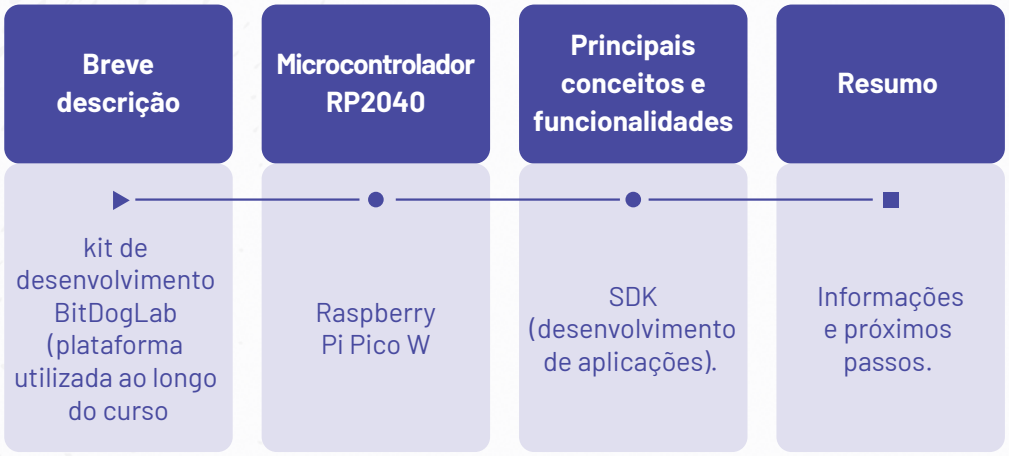
Além disso, esses sistemas interagem constantemente com o mundo físico, o que traz desafios adicionais. Restrições de tempo real, sincronização com sensores e atuadores, e a interface com sinais analógicos são aspectos que exigem atenção especial durante o desenvolvimento.

Outro ponto importante é que a **depuração** desses sistemas é frequentemente **complexa**, pois depende de sua interação com o ambiente físico ou com outros componentes, como motores e conversores de energia. Esses fatores tornam a depuração mais desafiadora, exigindo ferramentas e técnicas específicas para garantir o funcionamento correto do sistema.

Neste material de apoio, iremos apresentar o kit de microcontroladores e o SDK que serão utilizados ao longo do curso.
Vamos conhecer e utilizar o **kit de desenvolvimento BitDogLab**:

Raspberry Pi Pico W (uma das placas de microcontroladores mais populares atualmente) e sua plataforma educacional.

Ao longo do curso, exploraremos em detalhes o microcontrolador RP2040, que é o coração dessa placa, e aprenderemos a utilizar seu SDK para desenvolver aplicações práticas e avançadas em sistemas embarcados.



Nunca é demais lembrar: revisem todo o material deste capítulo com muita atenção e façam as atividades propostas na plataforma. É a sua chance de entender como os microcontroladores transformam o nosso dia a dia e aplicar esse conhecimento em projetos reais. Contamos com a sua participação ativa nessa jornada de aprendizado. Vamos em frente!

2. BitDogLab: uma ferramenta educacional

A **BitDogLab** é uma ferramenta educacional voltada para o ensino de sistemas embarcados, desenvolvida no contexto do projeto Escola 4.0 da Universidade de Campinas.

Essa ferramenta é construída com base na placa **Raspberry Pi Pico H** ou **W**, uma das mais populares no mundo. A BitDogLab se destaca, pois permite aos estudantes interagirem com um sistema embarcado baseado em microcontrolador. Ela proporciona um SDK que, embora fácil de aprender, é poderoso o suficiente para a criação de projetos complexos.

O kit vem equipado com um conjunto extenso de componentes, permitindo a realização de diversas práticas interativas, com uma rica

interface de usuário e a possibilidade de conectar módulos de expansão.

A placa inclui uma bateria de íon-lítio, o que possibilita seu funcionamento de forma autônoma, sem necessidade de conexão a uma tomada ou porta USB. Do ponto de vista da interface homem-máquina, a BitDogLab oferece um conjunto robusto de periféricos, como display gráfico, joystick, botões e uma matriz de LEDs. Além disso, a placa conta com um microfone integrado e um amplificador de áudio, ampliando as possibilidades de interação. Finalmente, conectores de expansão permitem que os estudantes adicionem sensores e/ou atuadores, tornando a BitDogLab uma ferramenta extremamente versátil para o aprendizado de sistemas embarcados. A figura abaixo apresenta o nosso kit de desenvolvimento, podemos observar os componentes mencionados no texto. A placa Raspbery Pi Pico W se encontra na parte posterior da placa.

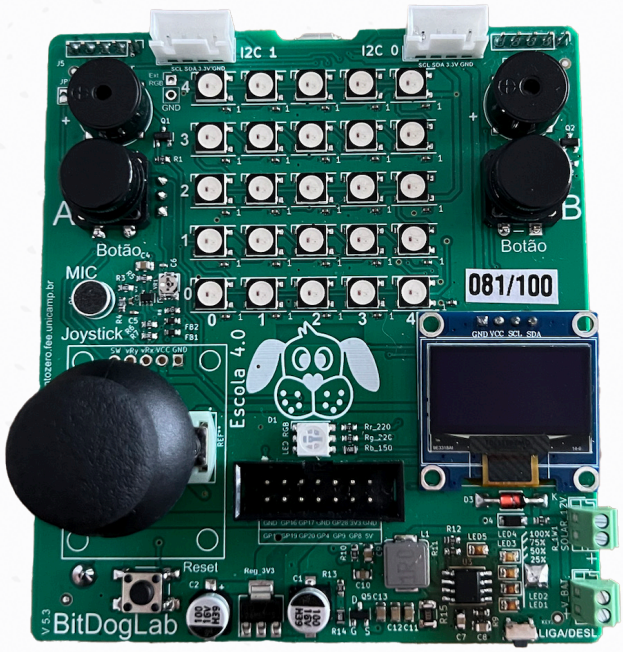


Figura 1: placa BitDogLab.
Fonte: https://github.com/BitDogLab/BitDogLab/tree/main/doc

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.

Como mencionamos anteriormente, o coração da **BitDogLab [1]** é a **Raspberry Pi Pico W**, uma solução de baixo custo e alta flexibilidade amplamente adotada em projetos de microcontroladores ao redor do mundo.
Essa placa é baseada no **microcontrolador RP2040 [2]**, que será detalhado mais adiante. Além disso, a Raspberry Pi Pico W possui interfaces de comunicação sem fio, incluindo **WiFi 802.11** e Bluetooth 5.2, o que a torna uma opção atrativa para aplicações em Internet das

Coisas (IoT).

O sistema também permite depuração in-circuit através da porta SWD, que requer o uso de um dispositivo externo para auxiliar no processo de depuração.

A Figura 2 ilustra a placa Raspberry Pi Pico W. Na imagem superior, é possível identificar o **microcontrolador RP2040**, o **botão BOOTSEL**, os **pinos de depuração**, e o **conector micro-USB**. Também há outros componentes importantes, como o **Infineon CYW43439**, que atua como controlador para os protocolos de comunicação sem fio.

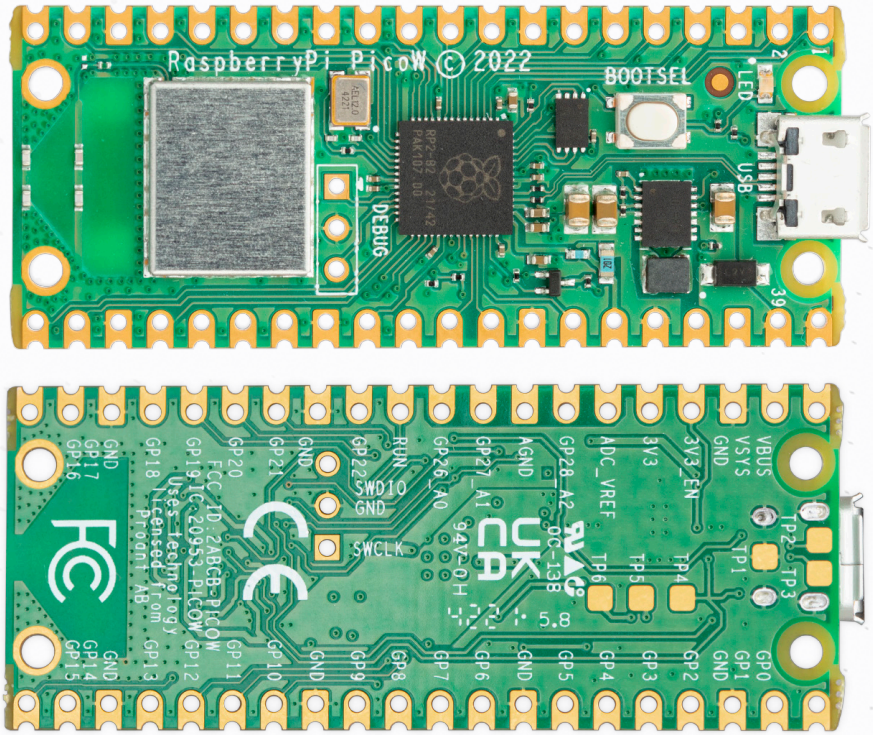


Figura 2: placa Raspberry Pi Pico W.
Fonte: https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf

A imagem mostra uma placa de desenvolvimento Raspberry Pi Pico W, com vistas da parte superior e inferior.
Vista Superior:
1.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.
2. 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.
3. Marcas:
- Na parte superior, lê-se "Raspberry Pi Pico W © 2022", indicando que é a versão com conectividade sem fio lançada em 2022.
Vista Inferior:
1. Pinos Numerados:
- A parte inferior da placa mostra a numeração dos pinos GPIO, com descrições indicando a função de cada um (como GND para terra, VBUS para alimentação, e GPIO com números para as entradas/saídas).
- O lado esquerdo e direito da placa têm os números dos pinos GPIO claramente marcados, facilitando a identificação.
2. Outras Marcas:
- Na parte inferior também estão impressas certificações como CE e FCC, indicando que o dispositivo está em conformidade com padrões de eletrônica e comunicação.
Essa placa é usada principalmente para projetos eletrônicos e de IoT (Internet das Coisas), com o diferencial da conectividade Wi-Fi.

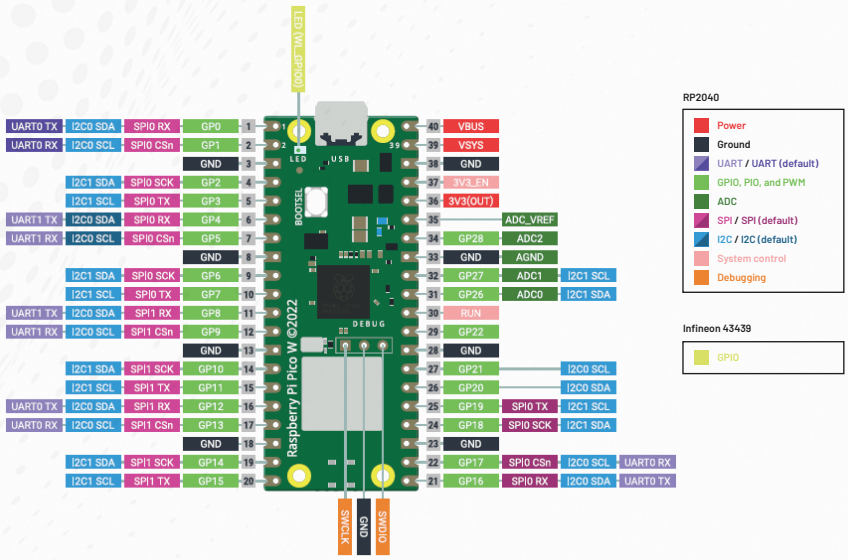


Figura 3: organização da placa.
Fonte: imagem do autor

A imagem mostra uma placa de circuito impresso (PCB) de um microcomputador chamado Raspberry Pi Pico W. É uma placa verde, pequena e retangular, com diversos componentes eletrônicos e conectores.
Elementos principais:
Formato: A placa tem formato retangular, com cantos arredondados.
Cor: A cor predominante é verde, mas há outros componentes em cores como preto, branco e cinza.
Componentes:
Conectores: Vários conectores metálicos estão dispostos ao longo das bordas da placa. Esses conectores são usados para conectar a placa a outros componentes eletrônicos, como sensores, atuadores e displays.
Chips: Há diversos chips integrados na placa, que são os "cérebros" da placa e realizam as funções de processamento.
Resistores e capacitores: Pequenos componentes eletrônicos, geralmente cilíndricos ou retangulares, usados para controlar o fluxo de corrente elétrica.
LEDs: Pequenas luzes indicadoras que acendem para mostrar o status da placa.
Legendas: Ao redor da placa, há diversas legendas indicando a função de cada conector e componente. Essas legendas estão em letras pequenas e em diferentes cores.
Legendas: Ao redor da placa, há diversas legendas indicando a função de cada conector e componente.
Essas legendas estão em letras pequenas e em diferentes cores.
Detalhes importantes: Conector USB: Um conector USB está localizado em uma das bordas da placa.
Ele é usado para conectar a placa a um computador para programação e alimentação. Botão de reset: Um pequeno botão está localizado próximo ao conector USB.
Ele é usado para reiniciar a placa. Antena: Uma pequena antena está conectada à placa, permitindo a conexão sem fio via Wi-Fi.

Não se preocupe, pois ao longo do curso iremos explorar cada um deles e os respectivos periféricos em detalhes!

Por enquanto, é suficiente saber que a placa possui 40 pinos, incluindo pinos de alimentação, depuração e outros dedicados aos periféricos, como GPIO, UART, SPI, I2C, ADC e PIO.

Na próxima seção, iremos apresentar alguns conceitos importantes relacionados ao microcontrolador RP2040, o cérebro dessa placa.

3. RP2040: um microcontrolador da Raspberry Pi.

O RP2040 [3] é um microcontrolador de baixo custo e alta flexibilidade, projetado para oferecer uma ampla gama de interfaces digitais versáteis. Ele é equipado com dois processadores ARM Cortex-M0+ que operam a até 133 MHz, além de 256 kB de memória RAM. O microcontrolador conta com 30 pinos de entrada e saída multifuncionais, uma porta para acesso a memória Flash externa, quatro canais analógicos, um módulo

PIO (Programmable Input/Output), e uma interface USB 1.1. Essas características fazem do RP2040 uma solução poderosa para diversas aplicações em sistemas embarcados.

A Figura 4 apresenta um diagrama de blocos simplificado que destaca os principais componentes da arquitetura interna do RP2040. Este microcontrolador possui dois núcleos de processamento ARM Cortex-M0+, que proporcionam um poder de processamento significativo para aplicações mais complexas. No entanto, durante nosso curso, iremos focar em aplicações que utilizam apenas um dos núcleos. O sistema de clock gera o relógio do sistema a partir de um cristal externo e um oscilador interno. O RP2040 também conta com um módulo DMA (Direct Memory Access), que acelera as transferências de dados entre periféricos e os bancos de memória. Os componentes do microcontrolador são interconectados por um barramento AHB/APB, organizado em dois níveis: o primeiro nível atende os componentes mais críticos, como os núcleos, DMA, memória RAM, acesso à Flash externa, PIO e USB; enquanto o segundo nível é destinado aos demais periféricos. O módulo GPIO tem conexão direta com os periféricos e com o módulo PIO, permitindo uma interface eficiente com o mundo externo.

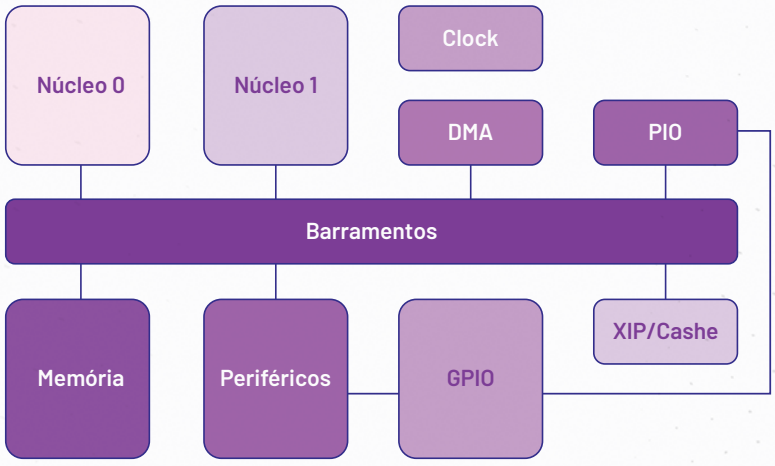


Figura 4: diagrama de blocos simplificado do RP2040.
Fonte: imagem do autor

A imagem mostra um diagrama de blocos representando a arquitetura interna de um microcontrolador, provavelmente o RP2040, utilizado no Raspberry Pi Pico. O diagrama destaca os principais componentes e a interconexão entre eles por meio de barramentos. A seguir estão os elementos descritos:
- Núcleo 0 e Núcleo 1: Representam os dois núcleos de processamento do microcontrolador. Eles são responsáveis pela execução de instruções e operam de forma paralela.
-Barramentos: Uma linha horizontal central conecta todos os blocos, indicando que os núcleos e periféricos se comunicam através desses barramentos.
- Memória: Bloco conectado aos barramentos, representando a memória onde o microcontrolador armazena dados e programas.
-Periféricos: Outro bloco conectado ao barramento, onde estão agrupados os dispositivos que permitem interações externas (sensores, comunicação, etc.).
-GPIO (General Purpose Input/Output): Bloco responsável pelos pinos de entrada e saída que permitem que o microcontrolador se conecte a dispositivos externos, como LEDs, botões e sensores.
- XIP/Cache: Indica a interface de execução em memória externa (XIP) e a memória cache, usada para otimizar o acesso a dados e instruções.
-CLOCK: Bloco que fornece o sinal de relógio para sincronizar todas as operações internas do microcontrolador.
-DMA (Direct Memory Access): Bloco que permite a transferência direta de dados entre a memória e periféricos, sem precisar envolver os núcleos de processamento.
- PIO (Programmable I/O): Bloco que permite a programação de pinos de entrada/saída para comportamentos específicos e personalizados, sem sobrecarregar os núcleos.
Esse diagrama destaca a organização modular do microcontrolador, mostrando como os diferentes componentes interagem para realizar tarefas de processamento e controle.

O RP2040 é equipado com uma ampla gama de periféricos que o tornam altamente atrativo para aplicações embarcadas. Ele inclui interfaces de comunicação serial, como UART, I2C e SPI, que permitem sua integração com outros processadores, sensores e atuadores. A interface PWM é ideal para o controle de motores e outros dispositivos que requerem modulação de largura de pulso. O microcontrolador também possui um módulo temporizador e um RTC (Real-Time Clock), que facilitam a criação de aplicações com restrições de temporização complexas. Além disso, o PIO (Programmable Input/Output) é um módulo versátil que permite emular interfaces de comunicação, criar barramentos personalizados, e até mesmo gerar áudio, ampliando ainda mais as possibilidades de uso do RP2040 em projetos inovadores.

4. O SDK da Raspberry Pi Pico W.

Os microcontroladores modernos, como o RP2040, possuem um hardware significativamente mais complexo do que os microcontroladores do início dos anos 2000. Essa complexidade advém da vasta gama de configurações possíveis, do grande número de periféricos integrados e das interfaces de comunicação sem fio. Configurar esses componentes diretamente, acessando os registradores manualmente, é uma tarefa complexa que pode facilmente levar a erros. Para simplificar esse processo, a Raspberry Pi fornece um SDK (Kit de Desenvolvimento de Software) [4] que inclui bibliotecas de baixo e alto nível. Essas bibliotecas abstraem a complexidade do hardware, oferecendo uma API fácil de usar, permitindo que os desenvolvedores configurem e utilizem os componentes do microcontrolador de forma mais intuitiva e segura.

Embora seja essencial que o desenvolvedor de software embarcado possua uma base sólida de conhecimento sobre o funcionamento dos periféricos e do microcontrolador, o uso de bibliotecas e APIs é altamente recomendável. Essas ferramentas não apenas aceleram o desenvolvimento das aplicações, mas também são componentes de software que foram amplamente testados e validados por desenvolvedores ao redor do mundo. Nesta seção, iremos introduzir os principais aspectos do SDK da Raspberry Pi Pico W. Este é um tema abrangente, que será explorado ao longo do curso. Para cada novo recurso ou periférico que estudarmos, será necessário também compreender quais funções, bibliotecas e APIs

nos auxiliam na configuração e utilização desse recurso.

O SDK oferece recursos para que programadores possam criar aplicações em C, C++ e Assembly. Neste curso, focaremos exclusivamente na criação de aplicações em linguagem C. O SDK disponibiliza as bibliotecas padrão do C, além de uma rica API para configurar e utilizar periféricos, gerenciar interrupções, DMA, e explorar recursos avançados como USB e programação multi-core. Além disso, o SDK inclui um sistema de build, que é uma ferramenta essencial para compilar e configurar nossas aplicações de maneira eficiente.

Vamos examinar como as bibliotecas do SDK estão organizadas. As bibliotecas de alto nível começam com 'pico_xxx', onde 'xxx' representa o nome da biblioteca. Por exemplo, a API que lida com funções avançadas de temporização é chamada de pico_time. As bibliotecas de baixo nível, que lidam diretamente com os periféricos e seus registradores, são nomeadas como 'hardware_xxx'. Além disso, existem bibliotecas que dão suporte ao ambiente de execução, configuração e inicialização do microcontrolador, garantindo que todos os componentes funcionem de maneira integrada e eficiente.

Ao longo do curso, iremos nos aprofundar nas funções e bibliotecas do SDK, mas para iniciarmos vamos fazer um exemplo simples. Para isso, você precisará já ter configurado o seu ambiente de desenvolvimento e ter uma BitDogLab em mãos. Em caso de dúvidas na configuração do seu ambiente ou programação da placa, reveja o material do Capítulo 4 da Unidade 3 ou procure nossa equipe pela plataforma educacional.

A Figura 5 apresenta o código da aplicação, que usa o pino GP12 para fazer piscar o led vermelho da placa BitDogLab. Esse programa consiste em um único arquivo fonte que contém a definição da função main(). No SDK, a função main() geralmente não possui parâmetros de entrada e, muitas vezes, não tem um valor de retorno, o que é comum em aplicações embarcadas.

No início do código, incluímos duas bibliotecas essenciais: a stdio.h, que é a biblioteca padrão de entrada e saída do C, e a pico/stdlib.h, que é uma biblioteca guarda-chuva. Essa biblioteca inclui várias outras bibliotecas comumente utilizadas para acessar os pinos de I/O, temporizadores,

entre outros recursos. Vale notar que as bibliotecas que começam com pico oferecem APIs de alto nível, enquanto as bibliotecas que começam com hardware normalmente são de baixo nível e lidam com configurações diretas de periféricos.

No código, definimos uma constante chamada led_pin_red, que tem o valor 12. Esse valor representa o pino GP12, que no caso da BitDogLab, está conectado ao LED vermelho do LED RDG. Dentro da função main, usamos as funções gpio_init() e gpio_set_dir() para configurar o pino GP12 como uma saída. Isso prepara o pino para controlar o LED. Em seguida, chamamos a função stdio_init_all(), que faz a configuração básica de entrada e saída, incluindo a configuração da porta serial. Isso permite que possamos usar as funções padrão do C para entrada e saída, como printf().

O código dentro do laço while(true) implementa a lógica para alternar o estado da saída do pino led_pin_red, fazendo o LED piscar. A função sleep_ms() é usada para introduzir um atraso no código, permitindo que o tempo passe entre as mudanças de estado do LED. Como resultado, o LED conectado ao pino GP12 piscará repetidamente, e a mensagem "Blinking!" será impressa na saída padrão a cada vez que o LED acender. Esse exemplo simples demonstra como configurar um pino GPIO como saída e como utilizar a função printf() para depuração ou comunicação com um console.

Um outro arquivo necessário para fazer uma aplicação SDK funcionar é o CMakeLists.txt, apresentado na Figura 6. Esse arquivo é responsável por configurar o processo de build do projeto, indicando quais arquivos devem ser compilados e linkados. Vamos dar uma olhada em como ele está organizado. Na primeira linha do CMake, utilizamos o comando cmake_minimum_required, que garante que a versão correta do CMake será usada para compilar o projeto. Isso é essencial para assegurar que todas as funcionalidades necessárias estejam disponíveis. Em seguida, fazemos a inclusão do caminho do SDK do Raspberry Pi Pico. Para isso, usamos o arquivo pico_sdk_import.cmake, que importa todas as configurações necessárias para compilar a aplicação. Isso inclui a configuração dos caminhos das bibliotecas e dos headers necessários para o desenvolvimento. Na próxima linha, definimos o nome do projeto com o comando project. No exemplo, o projeto é chamado pico-projects,

e indicamos que ele usará as linguagens C, C++, e Assembly, que são as linguagens suportadas pelo SDK do Pico.
Depois disso, iniciamos o SDK com o comando pico_sdk_init. Essa função garante que todas as bibliotecas e dependências estejam corretamente configuradas antes do início da compilação. A próxima etapa é adicionar o executável ao projeto com add_executable. Nesse exemplo, definimos o nome do executável como blink, que será gerado a partir do arquivo blink.c. Isso configura o CMake para compilar esse arquivo específico como o programa principal do projeto.

As bibliotecas necessárias são linkadas com o comando target_link_libraries. Esse passo é crucial, pois garante que teremos acesso às funções do GPIO, UART, e outras funcionalidades fornecidas pelo SDK do Pico. No exemplo, a biblioteca pico_stdlib é utilizada, que inclui funções básicas de entrada e saída. Por fim, indicamos que devem ser gerados os arquivos UF2 e BIN na saída da compilação com o comando pico_add_extra_outputs. Esses arquivos são os que serão efetivamente carregados no Raspberry Pi Pico para executar o programa.

```c
#include <stdio.h>
#include "pico/stdlib.h"
const uint led_pin_red = 12;
int main(){
  uint a = 100;

  gpio_init(led_pin_red);
  gpio_set_dir(led_pin_red, GPIO_OUT);
  stdio_init_all();
  while (true){
    a++;
    if (a % 2)
      printf("Blinking!\r\n");
    gpio_put(led_pin_red, true);
    sleep_ms(1000);
    gpio_put(led_pin_red, false);
    sleep_ms(1000);
  }
}
```

Figura 5. Arquivo CMakeLists.txt
Fonte: imagem do autor

Descrição da Imagem: Código em Linguagem C para Raspberry Pi Pico
Para uma pessoa com deficiência visual, a descrição de um código pode ser desafiadora, mas é possível transmitir a ideia geral do que o código faz.
Imagine um conjunto de instruções, como uma receita de bolo, mas escritas em uma linguagem específica que um computador entende. Esse conjunto de instruções é o que chamamos de código. No caso da imagem, o código está escrito na linguagem C, que é muito utilizada para programar microcontroladores como o Raspberry Pi Pico.
Vamos analisar o código passo a passo:
Inclusão de bibliotecas: As duas primeiras linhas, #include <stdio.h> e #include "pico/stdlib.h", são como se fossem receitas de bolo que você precisa consultar para fazer um determinado prato. Elas trazem para o nosso código um conjunto de funções prontas que facilitam a programação.
Definição de constantes: A linha const uint led_pin_red = 12; define uma constante chamada led_pin_red e atribui a ela o valor 12. Essa constante representa um pino específico no Raspberry Pi Pico que será usado para controlar um LED.
Função principal: A linha int main(){ marca o início da função principal do programa. É aqui que começa a execução do código.
Inicialização: As próximas linhas inicializam o pino do LED, a comunicação serial e uma variável auxiliar a.
Loop infinito: O while (true) { cria um loop que se repete infinitamente, ou seja, o código dentro desse loop será executado repetidamente.
Incremento e condição: Dentro do loop, a variável a é incrementada em 1 a cada repetição. Em seguida, é verificada se o valor de a é ímpar.
Ação: Se a for ímpar, o LED acende por 1 segundo, apaga por 1 segundo e uma mensagem é impressa na tela serial.
Repetição: Após executar as ações, o loop volta ao início e repete o processo.

```
cmake_minimum_required(VERSION 3.21)

include($ENV{PICO_SDK_PATH}/external/pico_sdk_import.cmake)

project(pico-projects C CXX ASM)

pico_sdk_init()

add_executable(blink
    blink.c
)

target_link_libraries(blink pico_stdlib)

pico_add_extra_outputs(blink)
```

Figura 6. Arquivo CMakeLists.txt
Fonte: imagem do autor

Claro, aqui está uma descrição da imagem:
A imagem mostra um código de computador em uma tela escura. O código está escrito em uma linguagem de programação chamada CMake. O código é usado para criar um projeto de software.
O código começa com uma linha que especifica a versão mínima do CMake necessária para executar o projeto. A próxima linha inclui um arquivo chamado "pico_sdk_import.cmake". Este arquivo contém código que é usado para importar o SDK do Raspberry Pi Pico.
O código então cria um projeto chamado "pico-projects". O projeto é escrito em C, C++ e ASM. O código então inicializa o SDK do Raspberry Pi Pico.
O código então adiciona um executável chamado "blink". O executável é criado a partir do arquivo "blink.c". O código então vincula o executável à biblioteca "pico_stdlib".
Finalmente, o código adiciona saídas extras ao executável.
Este código é um exemplo de como usar o CMake para criar um projeto de software para o Raspberry Pi Pico. O código é bem escrito e fácil de entender.

5. Conclusão

O desenvolvimento em microcontroladores é uma área empolgante e desafiadora. Microcontroladores modernos, como o RP2040, oferecem uma ampla gama de periféricos, tornando sua programação uma tarefa complexa. No entanto, o SDK da Raspberry Pi facilita essa tarefa, tornando o processo mais agradável e produtivo. Compreender os recursos de hardware e software do ecossistema de desenvolvimento da Raspberry Pi Pico W é essencial para sua jornada como desenvolvedor de aplicações embarcadas. Além disso, você tem à disposição um kit educacional especialmente projetado para auxiliá-lo nessa caminhada: a BitDogLab! Aproveite esta oportunidade e experimente replicar o exemplo de código apresentado na aula em sua placa.

Espero que este material de apoio seja útil e enriquecedor para o seu aprendizado. Por favor, aprofunde seus estudos por meio das referências deste texto, que servem como material suplementar. Vamos explorar juntos o fascinante mundo dos microcontroladores!

GLOSSÁRIO

802.11

A WiFi 802.11 é uma família de padrões de comunicação sem fio que permite a conexão de dispositivos em redes locais sem a necessidade de cabos.

Suas principais características incluem alta velocidade de transferência de dados, suporte a múltiplos dispositivos simultâneos e a capacidade de operar em diferentes bandas de frequência, como 2,4 GHz e 5 GHz.

O padrão 802.11 é amplamente utilizado em aplicações que exigem conectividade robusta e flexível, sendo essencial para a Internet das Coisas e outras tecnologias que dependem de comunicação sem fio eficiente.

Bluetooth 5.2

O Bluetooth 5.2 é uma versão avançada da tecnologia de comunicação sem fio Bluetooth, projetada para melhorar a eficiência e a flexibilidade das conexões entre dispositivos.

Suas principais características incluem maior velocidade de transmissão de dados, maior alcance, e melhor suporte para comunicação simultânea com múltiplos dispositivos.

Além disso, o Bluetooth 5.2 introduz melhorias no consumo de energia, tornando-o ideal para aplicações em dispositivos de baixo consumo, como wearables e sensores em sistemas de Internet das Coisas.

PIO

O PIO (Programmable Input/Output) é um módulo altamente flexível e poderoso do microcontrolador RP2040, projetado para realizar tarefas de entrada e saída que normalmente exigiriam hardware especializado ou sobrecarregariam o processador principal.

Com o PIO, é possível emular interfaces de comunicação personalizadas, como SPI, I2C, ou protocolos proprietários, criar sinais de temporização precisos, gerar padrões de comunicação complexos, e até mesmo produzir áudio.

O PIO opera de forma independente dos núcleos de processamento, permitindo que eles se concentrem em outras tarefas enquanto o PIO gerencia as operações de E/S em tempo real. Essa capacidade de programação e autonomia faz do PIO uma ferramenta versátil para desenvolvedores que necessitam de soluções de hardware personalizadas e eficientes.

SWD

O SWD (Serial Wire Debug) é um protocolo de depuração para microcontroladores, parte da arquitetura ARM Cortex. Ele oferece uma interface simples e eficiente para programar e depurar dispositivos embarcados.

Diferente de outras interfaces de depuração, como o JTAG, o SWD usa apenas dois fios para comunicação, o que reduz a complexidade e o custo do hardware.

Essa interface permite aos desenvolvedores acessarem a memória e os registradores do microcontrolador em tempo real, facilitando a identificação de problemas e a otimização do código durante o desenvolvimento de sistemas embarcados.

UART

O UART é um periférico que permite a comunicação serial entre dispositivos, transmitindo e recebendo dados de forma assíncrona. Ele converte dados paralelos em sinais seriais para transmissão e vice-versa na recepção, usando apenas dois fios: um para transmissão (TX) e outro para recepção (RX).

Ele é amplamente utilizado em sistemas embarcados para comunicação com outros dispositivos, como sensores, módulos de comunicação e outros microcontroladores.

I2C

O I2C é um protocolo de comunicação serial síncrono que permite a interconexão de múltiplos dispositivos em um único barramento de dois fios: SDA (Serial Data Line) e SCL (Serial Clock Line). Ele opera em uma arquitetura mestre-escravo, onde um único mestre pode se comunicar com vários escravos, transmitindo dados de forma eficiente. O I2C é amplamente utilizado para conectar periféricos de baixa velocidade, como sensores, relógios em tempo real e expansores de E/S.

SPI

O SPI é um protocolo de comunicação serial síncrono que permite a troca rápida de dados entre um microcontrolador e periféricos, como sensores, displays e memórias.

Ele utiliza uma arquitetura mestre-escravo, com quatro linhas principais: MISO (Master In Slave Out), MOSI (Master Out Slave In), SCK (Serial Clock), e SS (Slave Select).

O SPI é conhecido por sua alta velocidade e simplicidade, sendo ideal para aplicações que requerem transferência de dados rápida e confiável.

GPIO

O GPIO é um conjunto de pinos em um microcontrolador que podem ser configurados como entradas ou saídas digitais.

Esses pinos permitem a interface direta com outros componentes eletrônicos, como LEDs, botões e sensores. Os GPIOs são extremamente versáteis, podendo ser utilizados para ler estados lógicos ou controlar dispositivos externos, tornando-os fundamentais em quase todos os projetos de sistemas embarcados.

ADC

O ADC é um periférico que converte sinais analógicos, como os gerados por sensores de temperatura ou de luz, em valores digitais que podem ser processados por um microcontrolador.

A precisão do ADC é determinada por sua resolução, medida em bits, que define o número de níveis discretos em que o sinal analógico pode ser quantizado. O ADC é essencial para aplicações que envolvem a leitura de sinais do mundo real, permitindo que o microcontrolador intérprete e responda a essas entradas analógicas.

PWM

O PWM (Pulse Width Modulation) é um módulo essencial no microcontrolador RP2040, utilizado para gerar sinais de controle que variam em largura de pulso, permitindo o controle preciso de dispositivos como motores, LEDs, e outros atuadores.

Através da modulação da largura do pulso, o PWM pode controlar a potência entregue a um dispositivo, regulando, por exemplo, a velocidade de um motor ou o brilho de um LED.

Essa técnica é altamente eficiente, pois permite um controle contínuo e suave com baixo consumo de energia. O PWM é amplamente utilizado em sistemas embarcados para aplicações que requerem ajustes finos de energia e desempenho em tempo real.

DMA

O DMA (Direct Memory Access) é um módulo que permite a transferência direta de dados entre a memória e os periféricos, sem a intervenção contínua do processador.

Isso libera os núcleos de processamento para executar outras tarefas enquanto as transferências de dados ocorrem em segundo plano, aumentando a eficiência do sistema.

O DMA é especialmente útil em aplicações que envolvem grandes volumes de dados ou operações que exigem baixa latência, como a captura de dados de sensores, transmissão de dados em redes ou atualização de displays.

Ao reduzir a carga de trabalho do processador, o DMA contribui para um melhor desempenho geral do sistema e para a economia de energia.

API

Uma API (Application Programming Interface) é um conjunto de definições e protocolos que permite que diferentes softwares se comuniquem entre si. Ela atua como uma ponte, permitindo que aplicações acessem funcionalidades ou dados de outro sistema sem precisar conhecer os detalhes internos de sua implementação.

As APIs são essenciais no desenvolvimento moderno de software, facilitando a integração de serviços, o reuso de funcionalidades e a criação de aplicações mais complexas de forma eficiente e escalável. Por meio de APIs, desenvolvedores podem acessar recursos como bases de dados, serviços web, e bibliotecas de software, promovendo a interoperabilidade e a modularidade dos sistemas.

Referências

[1] BitDogLab. Manual BitDogLab, Disponível em: https://github.com/BitDogLab/BitDogLab/tree/main/doc Acesso em: 03 set. 2024.

[2] RASPBERRY PI LTD. Raspberry Pi Pico W Datasheet: An RP2040-based microcontroller board with wireless. 2024. Disponível em: https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf. Acesso em: 03 set. 2024.

[3] RASPBERRY PI LTD. RP2040 Datasheet. 2024. Disponível em: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf. Acesso em: 03 set. 2024.

[4] RASPBERRY PI LTD. Raspberry Pi Pico C/C++ SDK. 2024. Disponível em: https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-c-sdk.pdf. Acesso em: 03 set. 2024.