**Circuito eletrônico em fundo preto

Descrição gerada automaticamente com confiança médiaVoltBridgeADC**

**SHEILA RIBEIRO**

**Apresentação**

O VoltBridgeDAC é um sistema inovador de aquisição de sinais de tensão, desenvolvido com o objetivo de facilitar o acesso a tecnologias de medição e controle para pequenas empresas. Com uma solução de baixo custo, o VoltBridgeDAC busca superar as limitações dos equipamentos tradicionais, que frequentemente apresentam preços elevados, tornando-se inacessíveis para muitas empresas de menor porte.

A proposta do VoltBridgeDAC é oferecer uma alternativa acessível e eficiente, baseada em um microcontrolador STM32L053R8T6, que é responsável pelo processamento dos sinais de tensão adquiridos. A conversão do sinal analógico para digital é realizada por um circuito especializado, e a interface de comunicação com o usuário é simplificada através do software LabView, proporcionando uma interação intuitiva e de fácil operação.

Além de ser um sistema econômico, o VoltBridgeDAC se destaca pela sua precisão e versatilidade, sendo ideal para uma ampla gama de aplicações industriais, desde a medição de processos até o monitoramento de sinais em tempo real. Sua arquitetura modular permite adaptações conforme as necessidades do usuário, tornando-o escalável e flexível.

Com o VoltBridgeDAC, pequenas empresas poderão obter um controle mais eficiente de seus processos, sem o alto custo associado aos sistemas tradicionais de aquisição de dados, impulsionando a inovação, a competitividade e a capacitação tecnológica no mercado.

# **FUNCIONAMENTO**

## **DESCRIÇÃO**

O **VoltBridgeDAC** é um sistema de aquisição de sinais de tensão baseado em um microcontrolador **STM32L053R8T6**, que visa proporcionar uma solução eficiente e precisa para a medição e controle de sinais de tensão. O projeto é composto por um circuito eletrônico integrado ao microcontrolador e uma interface de comunicação com o usuário desenvolvida em **LabVIEW**. Essa interface permite que o usuário interaja com o dispositivo de forma intuitiva, monitorando e controlando os sinais de tensão em tempo real.

O **VoltBridgeDAC** foi projetado para ser altamente confiável e de fácil integração, tornando-o uma ferramenta valiosa para aplicações que exigem monitoramento de sinais elétricos em sistemas de automação e controle. Além disso, o projeto pode ser facilmente expandido ou modificado conforme as necessidades de diferentes aplicações.

## **STMCubeIDE**

O **STM32CubeIDE** é um ambiente de desenvolvimento integrado (IDE) robusto e completo, projetado especificamente para o desenvolvimento de aplicações em microcontroladores da família **STM32**, fabricados pela **STMicroelectronics**. Baseado na plataforma **Eclipse**, que é amplamente reconhecida no mercado, o STM32CubeIDE proporciona uma interface intuitiva e altamente funcional, combinando ferramentas de desenvolvimento, depuração e gerenciamento de configurações em um único ambiente. Este IDE oferece uma solução eficiente para engenheiros e desenvolvedores que buscam uma abordagem integrada e simplificada para o desenvolvimento de software embarcado.

O link contém o passo a passo de como baixar e explorar o software:

www.https://embarcados.com.br/como-instalar-o-stm32cubeide-guia-de-primeiros-passos/#Introducao

## **Programa em Linguagem C**

O código apresentado é o firmware de um microcontrolador STM32, desenvolvido em linguagem C utilizando a biblioteca HAL. Ele realiza a aquisição de sinais analógicos através de um ADC e comunica os valores obtidos via interface serial UART. O programa é projetado para operar continuamente, recebendo comandos e transmitindo dados formatados, sendo parte de um sistema maior voltado para a leitura e processamento de sinais de tensão.

#### **COMUNICAÇÃO SERIAL**

A comunicação serial é um método de transmissão de dados em que os bits são enviados sequencialmente, um após o outro, por um único canal de comunicação. Ao contrário da comunicação paralela, onde vários bits são enviados simultaneamente por canais separados, a comunicação serial utiliza apenas um ou dois fios para a transmissão (um para envio e outro para recebimento), o que a torna uma solução mais simples e econômica, especialmente para longas distâncias.

No contexto de sistemas embarcados e microcontroladores, a comunicação serial é fundamental para conectar dispositivos e trocar informações. Os dois principais canais usados para a comunicação serial nos microcontroladores são:

* **TX (Transmitter)**: O pino TX é responsável pela **transmissão** dos dados. Quando o microcontrolador envia dados para outro dispositivo, ele os envia através do pino TX, de forma sequencial, um bit por vez. O TX pode ser conectado ao RX de outro dispositivo para estabelecer a comunicação.
* **RX (Receiver)**: O pino RX é responsável pela **recepção** dos dados. Quando o dispositivo está recebendo dados de outro dispositivo, os bits chegam através do pino RX e são processados pelo microcontrolador.

**Figura 1**

**Gráfico, Gráfico de linhas

Descrição gerada automaticamente**

#### **COMUNICAÇÃO UART E SUA OPERAÇÃO ASSÍCRONA**

**O que é uma comunicação assíncrona**:

Forma

Descrição gerada automaticamente com confiança médiaEm uma comunicação **assíncrona**, como a que ocorre com a UART, não há necessidade de um sinal de relógio compartilhado entre os dispositivos que estão se comunicando. Em vez disso, os dispositivos sincronizam a transmissão e recepção de dados por meio de **taxas de transmissão fixas**, conhecidas como **baud rate**. Esse método de comunicação é chamado de "assíncrono" porque a transmissão de dados não depende de um relógio central compartilhado, ao contrário da comunicação síncrona, onde a sincronização de tempo é fundamental para garantir que os dados sejam enviados e recebidos de forma correta.

**Figura 2**

A comunicação assíncrona é extremamente eficiente em termos de hardware e recursos, pois não exige circuitos complexos para a sincronização do relógio entre os dispositivos. A principal responsabilidade dos dispositivos que se comunicam via UART é garantir que ambos estejam configurados para operar na mesma **taxa de transmissão** (baud rate), para que os dados sejam recebidos corretamente.

**Comunicação UART:**

Diagrama

Descrição gerada automaticamenteNa comunicação UART, os dados são enviados bit a bit de maneira sequencial, utilizando o canal TX do transmissor e o canal RX do receptor. Cada transmissão de dado é composta por uma série de bits, normalmente começando com um bit de início (start bit), seguido por 5 a 9 bits de dados, e finalizando com um bit de parada (stop bit), que indica o término da transmissão. A UART também pode incluir um bit de paridade para verificação de erros, garantindo a integridade dos dados.

**Figura 3**

A operação assíncrona na UART significa que a sincronização entre os dispositivos é realizada por meio da configuração de taxas de transmissão iguais em ambos os lados da comunicação. A partir dessa configuração, o transmissor envia os bits de forma sequencial, e o receptor interpreta os bits com base na sua taxa de baud, sem a necessidade de um sinal de relógio contínuo.

#### **BIBLIOTECA HAL**

A **biblioteca HAL (Hardware Abstraction Layer)** é uma das ferramentas mais importantes para o desenvolvimento de sistemas embarcados, especialmente quando se trabalha com microcontroladores, como os da família STM32. Ela fornece uma camada de abstração entre o software da aplicação e o hardware, simplificando a programação ao permitir que o programador se concentre nas funcionalidades de alto nível, sem se preocupar com os detalhes de baixo nível do hardware. Ao usar a biblioteca HAL, o código se torna mais portável, modular e fácil de manter, além de oferecer uma interface consistente para acessar os recursos de hardware do microcontrolador.

**O que é a biblioteca HAL?**

A HAL é uma biblioteca desenvolvida pela STMicroelectronics, projetada para facilitar a programação de microcontroladores STM32. Ela abstrai a complexidade da comunicação direta com os periféricos do microcontrolador, fornecendo funções de alto nível para manipulação de diversos módulos, como GPIO (General Purpose Input/Output), UART (Universal Asynchronous Receiver/Transmitter), I2C (Inter-Integrated Circuit), SPI (Serial Peripheral Interface), timers, ADC (Conversor Analógico para Digital), entre outros.

Ao utilizar a HAL, os desenvolvedores podem configurar e controlar o hardware do microcontrolador sem precisar acessar diretamente os registradores e os detalhes de cada periférico, tornando o código mais legível e menos propenso a erros.

**Como a Biblioteca HAL É Aplicada no Código:**

No código desenvolvido para o seu projeto, a biblioteca HAL tem um papel crucial na abstração da interação com os periféricos do microcontrolador STM32L053R8T6. Através dela, são realizadas operações essenciais para o funcionamento do sistema, como a configuração da UART para comunicação serial, o controle de GPIOs, a configuração de temporizadores e a manipulação de interrupções.

Por exemplo, a comunicação UART no seu código é realizada utilizando funções da HAL para configurar e gerenciar os canais TX e RX, responsáveis pela transmissão e recepção de dados, respectivamente. O uso da HAL simplifica a configuração dos parâmetros de comunicação, como o baud rate, os bits de paridade e a quantidade de bits de dados, sem a necessidade de programar esses parâmetros diretamente nos registradores do microcontrolador.

A HAL também facilita a utilização de outros recursos, como GPIOs para interações com o ambiente externo. Por exemplo, ao controlar LEDs ou botões, a biblioteca fornece funções para configurar os pinos como entradas ou saídas, facilitando o controle de hardware.

Além disso, a configuração de interrupções e o controle de temporizadores para tarefas de temporização são simplificados com a biblioteca HAL, permitindo a criação de sistemas multitarefa ou a execução de funções em intervalos específicos sem a necessidade de programar diretamente o hardware de interrupção.

#### **Linguagem C**

A **linguagem C** é uma das linguagens de programação mais fundamentais e amplamente utilizadas no desenvolvimento de sistemas embarcados, aplicativos de baixo nível e sistemas operacionais. Criada por Dennis Ritchie nos anos 70, ela se destaca pela sua simplicidade, eficiência e controle direto sobre o hardware. A linguagem C continua sendo uma das mais populares devido à sua portabilidade, flexibilidade e poder, sendo frequentemente usada para programar microcontroladores, como os da família STM32.

**Características da Linguagem C:**

A linguagem C é uma linguagem de programação estruturada e imperativa, o que significa que a execução do programa segue uma sequência de instruções lineares e lógicas. Ao longo das décadas, a C foi aprimorada e se consolidou como uma das principais linguagens de programação para sistemas que exigem alto desempenho e controle sobre os recursos de hardware.

**Regras de Sintaxe da Linguagem C:**

A sintaxe da linguagem C é baseada em regras rígidas que devem ser seguidas para que o código seja compilado corretamente e para que o comportamento do programa seja o esperado. Algumas das regras de sintaxe essenciais incluem:

1. **Uso de Pontuação (Delimitadores):**

* **Ponto e vírgula (;):** Em C, o ponto e vírgula é utilizado para **finalizar** instruções. Cada linha de código que realiza uma ação deve ser terminada com um ponto e vírgula, exceto em blocos de código dentro de estruturas de controle (como loops e funções).
* **Chaves ({ e }):** As chaves são usadas para agrupar um conjunto de instruções em blocos de código, como em funções, loops e estruturas condicionais. Elas definem o escopo do código que será executado dentro dessas estruturas.

1. **Declaração de Variáveis:**

* Antes de usar qualquer variável em C, ela deve ser **declarada** com um tipo de dado específico, como int, float, char, entre outros.
* As variáveis também precisam ser **inicializadas** para que possam ser usadas com valores válidos.

1. **Funções:**

* Em C, as funções são blocos de código que executam tarefas específicas. Para declarar uma função, deve-se especificar o tipo de dado retornado (como void para funções que não retornam valor), o nome da função e os parâmetros entre parênteses.

1. **Comentários**:

* Comentários são usados para explicar o código, tornando-o mais compreensível. Em C, os comentários podem ser de duas formas:
* **Comentário de linha única**: Usando //.
* **Comentário de múltiplas linhas**: Usando /\* para iniciar e \*/ para finalizar.

1. **Operadores**:

* A linguagem C oferece uma ampla gama de operadores para realizar operações matemáticas, lógicas e relacionais. Alguns exemplos incluem:
  + **Operadores aritméticos**: +, -, \*, /, %
  + **Operadores lógicos**: &&, ||, !
  + **Operadores relacionais**: ==, !=, <, >, <=, >=

1. **Tipos de Dados**:

* C suporta uma grande variedade de tipos de dados, incluindo tipos primitivos como int, char, float, e double. Além disso, permite a criação de tipos compostos, como **estruturas** (struct) e **uniões** (union).
  + Exemplo: float altura = 1.75;

1. **Estruturas Condicionais e de Repetição**:

* **Condicionais**: Usadas para tomar decisões dentro do código. As principais são if, else e switch.
* **Loops**: Usados para repetir um bloco de código. As principais estruturas de repetição são for, while e do-while.

**Boas Práticas e Convenções**

Além da sintaxe, a linguagem C também segue certas convenções que ajudam na legibilidade e manutenibilidade do código. Algumas boas práticas incluem:

* **Indentação**: A indentação adequada do código é fundamental para a organização do código-fonte. Embora C não exija um formato específico, a prática comum é usar espaços ou tabulações para indicar blocos de código.
* **Nomeação de Variáveis**: Usar nomes de variáveis que descrevem claramente sua função ou valor é importante para a legibilidade do código. Por exemplo, usar contador ao invés de x em um loop.
* **Evitar o uso de "números mágicos"**: É recomendável declarar constantes para valores numéricos, em vez de usá-los diretamente no código, o que facilita a manutenção do sistema.

#### **O Código em Linguagem C**

Este código foi o programa em C para um microcontrolador STM32, utilizando a biblioteca HAL (Hardware Abstraction Layer) da STMicroelectronics. Ele configura e utiliza diversos periféricos do microcontrolador, incluindo o ADC (Conversor Analógico-Digital) e UART (Comunicação Assíncrona Universal), com o objetivo de ler dados do ADC e transmitir resultados via UART. Parte do programa foi criado automaticamente pelo software STMCubeIDE. A seguir, a explicação do código será dividida em suas partes principais

1. **Cabeçalho do Arquivo**

Essa seção é um cabeçalho padrão gerado pela STMicroelectronics. Ele contém informações de copyright, a licença do software e uma descrição do arquivo.

Texto

Descrição gerada automaticamente**Figura 4**

1. **Inclusões e Definições**

Aqui, o arquivo de cabeçalho main.h é incluído, que provavelmente contém as definições necessárias para o funcionamento do código, como declarações de variáveis globais e funções de inicialização.

**Figura 5**

**Interface gráfica do usuário

Descrição gerada automaticamente**

**3**. **Definição de Variáveis e Structs**

Texto

Descrição gerada automaticamente com confiança média**Figura 6**

* hadc: Instância do manipulador para o ADC.
* huart2: Instância do manipulador para a comunicação UART.
* value: Variável para armazenar o valor lido do ADC.
* rxdata: Array para armazenar os dados recebidos via UART.
* a: Contador simples para controle dentro do loop.
  1. **Prototipagem de Funções**

Aqui são declaradas as funções auxiliares para configurar o sistema de clock, os periféricos GPIO, USART2 (UART) e ADC.

**Figura 7**

Interface gráfica do usuário

Descrição gerada automaticamente

* 1. **Definição do PUTCHAR\_PROTOTYPE para Impressão via UART**

Este trecho define uma função de protótipo PUTCHAR\_PROTOTYPE para a função de impressão via UART. Dependendo do compilador utilizado (no caso, o GCC), a implementação da função será diferente. Isso permite usar printf para enviar dados pela UART.

Texto

Descrição gerada automaticamente**Figura 8**

* 1. **Função \_\_io\_putchar para Transmissão de Dados via UART**

Texto

Descrição gerada automaticamenteEsta função envia um caractere via UART usando a função HAL\_UART\_Transmit, que é fornecida pela biblioteca HAL. Essa implementação é necessária para permitir o uso de funções como printf.

**Figura 9**

* 1. **Interface gráfica do usuário, Texto, Aplicativo

     Descrição gerada automaticamenteFunção main (Ponto de Entrada do Programa)**

**Figura 10**

* Inicialização: A função HAL\_Init() é chamada para inicializar a biblioteca HAL. Em seguida, o sistema de clock e os periféricos GPIO, UART e ADC são configurados.
* Loop Principal: O programa entra em um loop infinito (while(1)), onde:
* Recebe dados via UART (HAL\_UART\_Receive).
* Inicia a conversão do ADC, aguarda a conversão ser completada (HAL\_ADC\_PollForConversion), obtém o valor do ADC (HAL\_ADC\_GetValue), e então faz uma pequena pausa (HAL\_Delay).
* O valor lido do ADC é impresso em formato específico dependendo do seu valor.
* O if (rxdata[0] == 's') verifica se o comando recebido via UART é a letra 's', e caso seja, o valor do ADC é formatado e enviado via printf.

1. **Função de Configuração do Clock do Sistema**

Interface gráfica do usuário

Descrição gerada automaticamente com confiança médiaEsta função configura o relógio do sistema, incluindo a calibração do oscilador, o tipo de oscilador, a divisão de clocks, e a configuração do PLL (se necessário). Ela é necessária para garantir que o microcontrolador funcione com a frequência e as condições de clock adequadas.

**Figura 11**

1. **Função de Inicialização do ADC**

Esta função inicializa o ADC, configurando características como resolução, alinhamento de dados, e modo de conversão. O canal do ADC também é configurado.

Logotipo, nome da empresa

Descrição gerada automaticamente**Figura 12**

1. **Função de Inicialização do UART**

Logotipo

Descrição gerada automaticamente com confiança baixaEsta função configura a UART2 para comunicação, definindo parâmetros como taxa de transmissão (baud rate), número de bits de dados, bits de parada, paridade e controle de fluxo.

**Figura 13**

1. **Função de Inicialização do GPIO**

Logotipo

Descrição gerada automaticamente com confiança baixaConfigura os pinos GPIO necessários para o funcionamento do sistema, permitindo a comunicação UART e outros periféricos.

**Figura 14**

1. **Função de Tratamento de Erro**

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamenteEm caso de erro, o programa entra em um loop infinito, impedindo a execução de outras tarefas e bloqueando a execução.

**Figura 15**

## **Circuito**

O circuito pode ser dividido em três partes, sendo elas: entrada USB e alimentação; microcontrolador; e aquisição de dados.

#### **ENTRADA E ALIMENTAÇÃO**

**Figura 16**

**Diagrama, Esquemático

Descrição gerada automaticamente**

**Figura 17**

Cabo preto em fundo branco

Descrição gerada automaticamente

A entrada USB é composta por um módulo **USB\_b\_Micro**, que fornece seis canais distintos, a saber:

* **VBUS**: Canal responsável pela alimentação do circuito com 5 V.
* **D+**: Canal de dados positivos.
* **D-**: Canal de dados negativos.
* **ID**: Canal de identificação do dispositivo.
* **GND**: Canal de referência de terra.
* **Shield**: Canal de blindagem, utilizado para proteger os sinais transmitidos pelos pinos D+ e D- contra interferências eletromagnéticas externas.

Este módulo USB é seguido pelo dispositivo **USBLC6**, que é um **protetor de linha de dados**. O USBLC6 oferece proteção eficaz contra surtos de tensão, com capacidades de até **±15 kV** em descarga de contato e **±25 kV** em descarga de ar. Este componente é projetado para proteger os dados em ambas as direções, tanto para surtos positivos quanto negativos, e é compatível com a especificação **USB 2.0** de alta velocidade (40 Mbps), além de atender à norma **IEC 61000-4-2** para testes de compatibilidade eletromagnética (EMC).

O **USBLC6** é um dispositivo passivo e não requer alimentação externa para operação. Sua aplicação é ideal em sistemas que exigem proteção de linhas de dados USB, assegurando a integridade dos sinais em ambientes com alto risco de interferência.

**Distribuição dos canais de proteção do USBLC6:**

* **Canal 1**: Protege a linha de dados **D+** contra sobretensões e surtos de tensão eletroestática (ESD).
* **Canal 2**: Protege a linha de **terra (GND)** contra surtos de ESD.
* **Canal 3**: Protege linhas de dados adicionais, como as utilizadas por dispositivos de alta velocidade com tecnologia **ULPI** (USB Low Pin Interface), contra sobretensões e surtos de ESD.
* **Canal 4**: Protege a linha de dados **D-** contra sobretensões e surtos de ESD.
* **Canal 5**: Protege a linha de alimentação **VBUS (5V)** contra surtos de ESD.
* **Canal 6**: Protege uma linha de dados adicional, como a linha **D+** em dispositivos USB 2.0, contra sobretensões e surtos de ESD.

Todos os resistores e capacitores do circuito, incluindo o **resistor de ferrite beads** (FB101), trabalham em conjunto para minimizar possíveis interferências e ruídos, garantindo a estabilidade e o desempenho do sistema.

Essa configuração robusta de proteção e filtragem assegura que o sistema opere de maneira confiável, mesmo em ambientes sujeitos a distúrbios eletromagnéticos e transientes de tensão.

#### **Diagrama, Esquemático Descrição gerada automaticamenteMICROCONTROLADOR**

**Figura 18**

****

**Figura 19**

O microcontrolador utilizado é o **STM32L053R8T6**, da STMicroelectronics, projetado para operar com consumo de energia extremamente baixo, enquanto oferece alto desempenho com conectividade USB 2.0 (sem cristal) e o núcleo **Arm Cortex-M0+**. Este processador RISC de 32 bits funciona a uma frequência de 32 MHz e conta com uma Unidade de Proteção de Memória (MPU), garantindo a segurança e integridade dos dados.

O **STM32L053R8T6** é equipado com diversos recursos analógicos, incluindo um ADC de 12 bits com oversampling por hardware, que proporciona alta precisão na conversão de sinais. Além disso, oferece um conjunto robusto de interfaces de comunicação padrão e avançadas, como até dois **I2C**, dois **SPI**, um **I2S**, dois **USART**, um **UART** de baixa potência (**LPUART**) e um **USB** sem a necessidade de cristal.

Como integrante da linha de microcontroladores de **ultrabaixa potência** da família **STM32L053x6/8**, o STM32L053R8T6 opera com uma faixa de alimentação de 1,8 a 3,6 V (com capacidade de operar até 1,65 V em modo de desligamento). A ampla gama de **modos de economia de energia** permite a otimização do consumo, tornando-o ideal para aplicações que exigem eficiência energética sem comprometer o desempenho.

#### **ALIMENTAÇÃO DO MICROCONTROLADOR**

A alimentação do microcontrolador pode ser fornecida de duas fontes distintas, dependendo da disponibilidade de energia: **VBUS**, que provém da interface **USB**, e **+VBAT**, que é a alimentação proveniente de uma **bateria**.

* **VBUS (alimentação via USB)**: Quando o dispositivo está conectado a uma porta USB, o **VBUS** fornece a alimentação de **5V**. O padrão USB permite que o sinal de alimentação seja transmitido através da linha **VBUS**, oferecendo uma solução conveniente para fornecer energia ao sistema. Este tipo de alimentação é amplamente utilizado em dispositivos que podem ser conectados diretamente a um computador ou a um carregador USB, dispensando a necessidade de fontes externas adicionais.
* **+VBAT (alimentação via bateria)**: Quando o dispositivo não está conectado a uma fonte USB ou quando a alimentação via USB não está disponível, o sistema pode ser alimentado por uma **bateria**. A entrada **+VBAT** é conectada a uma fonte de energia de **tensão contínua**, tipicamente uma célula de bateria, que garante o funcionamento do microcontrolador e do sistema de aquisição de dados em condições de operação autônoma. Este tipo de alimentação é útil em aplicações móveis ou em sistemas que exigem portabilidade, onde uma alimentação via USB não seria viável ou disponível.

A alimentação do sistema é gerenciada pelo **PWM Source**, que seleciona automaticamente a fonte de energia disponível entre **VBUS** e **+VBAT**. Quando o dispositivo está conectado a uma porta USB, a alimentação é fornecida por **VBUS**, enquanto, na ausência de USB, o circuito comuta para a **bateria** através de **+VBAT**. Independentemente da fonte de alimentação selecionada, a saída será sempre **VDD**, com uma tensão de 5V, garantindo uma operação estável e contínua. Esse controle automático assegura que o sistema opere sem interrupções, utilizando a fonte de energia mais adequada conforme a disponibilidade.

Diagrama, Esquemático

Descrição gerada automaticamente

**Figura 20**

A tensão **VDD** de 5V é direcionada ao regulador de tensão linear **LM317**, um dispositivo ajustável de três terminais, projetado para fornecer até 1,5 A de corrente com uma faixa de saída ajustável de **1,25 V a 37 V**. Este regulador requer apenas dois resistores externos para definir a tensão de saída, o que oferece flexibilidade para aplicações com diferentes requisitos de alimentação.

Uma imagem contendo Diagrama

Descrição gerada automaticamente

**Figura 21**

Como ilustrado na **Figura 7**, a saída do regulador de tensão **VDD\_3V3** é utilizada para alimentar os pinos **VDD**, **VDDA** e **VREF** do microcontrolador. O **VDD\_3V3** passa por um dos terminais de **capacitores de filtragem**, enquanto o outro terminal é conectado ao **terra (GND)**. Em seguida, os sinais de alimentação filtrados são direcionados às entradas correspondentes no microcontrolador, garantindo uma alimentação estável e livre de ruídos para o funcionamento dos módulos analógicos e digitais.

**Figura 22**

Tela de computador com fundo verde

Descrição gerada automaticamente com confiança média

#### **CRISTAL OSCILADOR**

O microcontrolador **STM32L4R9AII6** é equipado com dois pinos dedicados à entrada de cristal oscilador, denominados **OSC\_IN** e **OSC\_OUT**. Esses pinos são utilizados para a conexão de um cristal externo ou um oscilador externo, proporcionando uma referência de frequência precisa para o microcontrolador. Este recurso permite a obtenção de uma frequência de operação estável e confiável, essencial para aplicações que demandam alta precisão temporal.

O microcontrolador **STM32L4R9AII6** utiliza o cristal oscilador **NX3225GA**, um cristal de quartzo que opera a uma frequência de **16 MHz**, fornecendo uma referência de frequência precisa para o sistema. Para garantir a estabilidade e o desempenho do sinal de oscilação, são utilizados capacitores de filtragem **CC0603CRNPO9BN6R8**, com valor nominal de **9 pF** e uma tensão máxima de **50 VDC**.

Diagrama, Linha do tempo

Descrição gerada automaticamente com confiança média

**Figura 23**

Linha do tempo

Descrição gerada automaticamente com confiança médiaAlém dos pinos **OSC\_IN** e **OSC\_OUT**, que se conectam ao cristal oscilador, o microcontrolador também utiliza os pinos **OSC32\_IN** e **OSC32\_OUT** para alimentar o módulo **Real-Time Clock (RTC)**. Este módulo dedicado permite a manutenção do controle de tempo e data mesmo quando o microcontrolador está em modos de baixo consumo ou desligado, assegurando que o sistema preserve a referência temporal em todas as condições de operação, incluindo em situações de desligamento ou economia de energia.

**Figura 24**

#### **AQUISIÇÃO DE DADOS**

O circuito de aquisição de dados é composto por um amplificador operacional configurado como somador não inversor, alimentado por **+12V** e **-12V**. Este circuito apresenta ganho unitário e sua saída é conectada a um divisor de tensão, permitindo a adequada redução da amplitude do sinal para os níveis desejados. A configuração garante uma amplificação precisa e estável dos sinais de entrada, com controle eficiente dos níveis de tensão para o processamento subsequente dos dados.

#### **FONTE SIMÉTRICA**

A alimentação do amplificador operacional (ampop) é proveniente de uma fonte simétrica com entrada de **24V**, a qual é dividida em **+12V** e **-12V** para fornecer a alimentação adequada ao circuito. A divisão de tensão é realizada pelos **capacitores eletrolíticos** de **1000µF** e **35V**, que asseguram a estabilização das tensões positivas e negativas. Para a filtragem da alimentação, são utilizados **capacitores cerâmicos** de **220nF**, com uma tensão superior a **20V**, garantindo a remoção de ruídos e a estabilidade do sinal.

A proteção contra variações de tensão é realizada pelos **diodos Zener** de **12V**, que impedem que a tensão de alimentação ultrapasse os limites especificados, garantindo a segurança e o funcionamento adequado do circuito. Próximos ao amplificador operacional, os **capacitores cerâmicos** de **100nF** (com tensão acima de **20V**) são empregados para filtrar e estabilizar a alimentação diretamente nas entradas do ampop. Esses capacitores estão conectados aos terminais de alimentação **+V ampop** e **-V ampop**, direcionando as tensões positivas e negativas para as respectivas entradas de alimentação do amplificador operacional.

Diagrama, Esquemático

Descrição gerada automaticamenteEsse arranjo assegura a operação eficiente do amplificador, com proteção, filtragem e estabilização adequadas para a aquisição de dados precisa.

**Figura 25**

#### **AMPLIFICADOR OPERACIONAL SOMADOR-NÃO INVERSOR**

O amplificador operacional utilizado neste projeto é o **LM741**, um componente versátil de alto desempenho. Este ampop é alimentado com tensões simétricas de **±12V**, respeitando sua alimentação máxima de **±22V**, e possui quatro pinos principais, que são:

* **OFFSET NULL**: Não utilizado neste projeto;
* **Inverting input**: Entrada inversora, utilizada para o ganho unitário do circuito;
* **Non-inverting input**: Entrada não inversora, onde são aplicadas as tensões **Vref** e **Vsensor**;
* **V-**: Alimentação negativa de **-12V**;
* **V+**: Alimentação positiva de **+12V**;
* **NC**: Pino não utilizado neste projeto;
* **Output**: Saída de tensão, com uma variação de **0V a ±11V**, dependendo das tensões de entrada

O LM741 é configurado como amplificador operacional **somador não-inversor**, sendo utilizado para realizar a soma ponderada de duas tensões de entrada, comparando-as e gerando uma saída proporcional. No contexto deste projeto, as duas tensões de entrada são **Vref**, a tensão de referência, e **Vsensor**, a tensão proveniente do sensor.

onde:

* **RA** e **RB** são os resistores de feedback e entrada, respectivamente, e **Q** é o número de resistores no divisor de tensão (considerando-se resistores com valores iguais).

Como o ganho do circuito é **unitário** (1), os resistores **RA** e **RB** podem ser descartados na equação. Portanto, a equação simplifica para:

***Observação: a quantidade de resistores como divisor, só deve ser considerada quando os resistores possuírem valores iguais.***

***Exemplos de Funcionamento:***

1. **Caso 1 - Vsensor = -12V e Vref = 12V:**
2. **Caso 2 - Vsensor = 0V e Vref = 12V**:
3. **Caso 3 - Vsensor = 10V e Vref = 12V:**

Este arranjo de amplificador operacional somador não inversor é fundamental para a aquisição e processamento de sinais de sensores, com a combinação de tensões **Vref** e **Vsensor** sendo comparada e processada de forma precisa. A configuração de ganho unitário assegura que a variação de **Vsensor** seja refletida diretamente na saída, enquanto a **Vref** contribui com um valor fixo que define o ponto de operação do circuito. Esse comportamento é crucial para a análise e interpretação dos sinais adquiridos em sistemas de medição e controle.

Diagrama, Esquemático

Descrição gerada automaticamente

**Figura 26**

Como ilustrado na **Figura 11**, o circuito de aquisição de dados é formado pelas tensões **Vsensor** e **Vref**, ambas conectadas a resistores de **10 kΩ**. Esses resistores definem as condições de entrada do amplificador operacional, com as alimentações já filtradas por capacitores de **100nF**. Na saída do amplificador operacional (ampop), três resistores são utilizados como um divisor de tensão, ajustando o nível da tensão para que esteja dentro dos limites aceitáveis pelo microcontrolador.

**Divisão de Tensão para Proteção do Microcontrolador:**

O **microcontrolador STM32L053R8T6** opera com uma tensão máxima de **3,6V** em seus pinos de entrada. Portanto, a saída do ampop, que pode ser maior que esse valor, precisa ser dividida para garantir que o microcontrolador não seja danificado. A saída do divisor de tensão, designada como **ADC output**, é então alimentada no pino de entrada **ADC** do microcontrolador.

**Seleção dos Pinos ADC:**

Os pinos **ADC** do STM32L053R8T6 selecionados para este circuito são **PA0**, **PA1**, **PA4** e **PA5**. No entanto, neste projeto específico, somente o pino **PA1** será utilizado para receber a entrada da tensão de saída do divisor de tensão. Os outros pinos estão conectados a conectores para permitir a entrada de tensões externas, possibilitando futuras expansões ou medições adicionais.

**Considerações Técnicas:**

* O uso de **resistores de 10 kΩ** e capacitores de **100nF** contribui para a estabilidade do circuito e a filtragem de ruídos de alta frequência, garantindo medições precisas.
* O divisor de tensão ajusta a saída do ampop para níveis compatíveis com a entrada de **3,6V** do microcontrolador, protegendo os pinos do ADC contra sobrecarga e possíveis danos devido a tensões excessivas.
* A escolha dos pinos **PA1**, **PA4**, **PA5** e **PA0** como entradas **ADC** permite flexibilidade no projeto, pois possibilita a utilização de pinos adicionais para futuras medições ou entradas externas.

Esta configuração assegura a correta operação do sistema, mantendo as entradas do microcontrolador dentro dos parâmetros de segurança, enquanto permite a medição precisa das tensões no circuito.

# **PCB**

## **DESCRIÇÃO**

A **Placa de Circuito Impresso (PCB)** é um componente fundamental no design e fabricação de dispositivos eletrônicos. Ela serve como a base física e a plataforma para conectar e interligar os diversos componentes eletrônicos, como resistores, capacitores, transistores e microcontroladores, de forma organizada e funcional. A PCB é composta por uma camada de material isolante, geralmente feita de fibra de vidro ou resina epóxi, sobre a qual são aplicadas camadas de cobre que formam as trilhas e conexões elétricas.

As **PCBs** são essenciais para a construção de praticamente todos os dispositivos eletrônicos modernos, desde os mais simples, como controles remotos, até os mais complexos, como computadores e smartphones. Elas são projetadas com precisão para garantir que os componentes funcionem corretamente, minimizando interferências e maximizando a eficiência elétrica.

Existem diferentes tipos de PCBs, incluindo **monocamada**, **bicamada** e **multicamada**, que variam em termos de complexidade e capacidade de acomodar circuitos mais densos. Além disso, a PCB pode ser projetada para atender a requisitos específicos de resistência a altas temperaturas, flexibilidade ou impermeabilidade, dependendo da aplicação.

O processo de fabricação de uma PCB envolve diversas etapas, como a impressão das trilhas de cobre, a colocação de componentes eletrônicos e a soldagem, tudo realizado com precisão para assegurar a qualidade e o desempenho do circuito. A confiabilidade, durabilidade e custo de uma PCB dependem de fatores como o material utilizado, o número de camadas, o design e as técnicas de fabricação empregadas.

## **O MATERIAL DA PCB**

O material de base de uma PCB afeta diretamente as características elétricas e mecânicas do circuito, incluindo a resistência térmica, a rigidez dielétrica, a capacidade de dissipação de calor, a estabilidade ao longo do tempo e a facilidade de fabricação. Para garantir o desempenho ideal do circuito e a integridade da placa, a escolha do material deve ser feita com base nas condições ambientais e nas exigências específicas do projeto, como a temperatura operacional, a frequência do sinal e a densidade de componentes.

Além disso, a resistência mecânica e a flexibilidade do material são determinantes para a durabilidade da placa, especialmente quando ela será submetida a vibrações ou outras condições adversas. A condutividade térmica também é um fator essencial, pois materiais com boa dissipação de calor garantem que componentes eletrônicos, como processadores e transistores, operem dentro de limites seguros de temperatura, evitando danos ao longo do tempo.

#### **Escolha do FR-4 para o Projeto VoltBridgeADC:**

Para o projeto **VoltBridgeADC**, a escolha do **FR-4** foi baseada em uma série de fatores que atendem de maneira eficaz às necessidades específicas da aplicação. O FR-4 é amplamente utilizado na indústria de eletrônicos devido às suas características balanceadas, incluindo:

1. **Custo-Benefício**: O FR-4 é um material econômico, o que torna o processo de fabricação mais acessível, especialmente em produções em grande escala. Sua ampla disponibilidade e custo reduzido fazem com que seja uma escolha ideal para projetos que buscam um bom desempenho sem comprometer o orçamento.
2. **Propriedades Elétricas Estáveis**: O FR-4 possui uma excelente rigidez dielétrica, o que assegura uma boa isolação elétrica entre as trilhas e os componentes da placa. Isso é fundamental para garantir o correto funcionamento do VoltBridgeADC, evitando curtos-circuitos ou interferências indesejadas.
3. **Resistência Térmica Adequada**: Para o VoltBridgeADC, que pode envolver circuitos de precisão e sensibilidade térmica, o FR-4 oferece boa resistência ao calor, o que permite que o circuito opere dentro da faixa de temperatura padrão sem risco de falhas térmicas. Além disso, a capacidade de dissipação de calor é suficiente para aplicações de baixo e médio poder.
4. **Durabilidade e Robustez**: O FR-4 é um material que combina rigidez e resistência mecânica, o que proporciona durabilidade à placa. Esse fator é essencial para garantir que o VoltBridgeADC, em suas aplicações de aquisição de sinais de tensão, opere de forma confiável em ambientes exigentes.
5. **Facilidade de Fabricação**: O FR-4 é amplamente utilizado na indústria de PCB, o que facilita a fabricação em termos de processos de perfuração, roteamento e soldagem. Isso proporciona uma produção mais rápida e precisa, além de garantir maior disponibilidade de fabricantes e fornecedores.

## **ACABAMENTO DA PCB E OPÇÕES TÉCNICAS**

O acabamento superficial de uma PCB é uma etapa essencial no processo de fabricação, desempenhando um papel fundamental na proteção das superfícies de cobre contra oxidação e outros agentes ambientais. Além disso, ele garante uma soldagem eficiente e confiável, tanto para componentes PTH (Plated Through Hole) quanto para SMD (Surface Mount Device). O acabamento é aplicado após a fabricação das trilhas e furos da placa, servindo como uma camada de proteção e facilitando a montagem dos componentes eletrônicos.

#### **FUNÇÕES DO ACABAMENTO SUPERFICIAL**

* **Proteção contra Oxidação**: Evita a deterioração das trilhas de cobre, mantendo a condutividade elétrica ao longo do tempo.
* **Melhoria da Soldabilidade**: Proporciona uma superfície uniforme e aderente para a solda, essencial para a formação de conexões elétricas seguras e confiáveis.
* **Estabilidade Mecânica e Térmica**: Garante que a placa suporte processos de montagem e ciclos térmicos durante sua vida útil.

#### **TIPOS DE ACABAMENTO**

**1. HASL (Hot Air Solder Leveling)**

O HASL é um dos acabamentos mais amplamente utilizados devido ao seu custo acessível e eficiência para aplicações de uso geral. Ele consiste na imersão da placa em solda estanho-chumbo fundida (ou estanho-sem chumbo para a versão Lead-Free), geralmente a uma temperatura de 260°C, seguida pela remoção do excesso de solda com facas de ar quente.

**Características Técnicas**:

* Boa proteção contra oxidação.
* Adequado para componentes convencionais e SMD em placas com baixa densidade de componentes.
* Espessura da camada de solda pode variar, reduzindo a uniformidade para componentes de alta precisão.
* Custo: Baixo.

**Recomendações**: Ideal para protótipos e aplicações que não exigem tolerâncias extremas ou alta densidade de componentes.

**2. ENIG (Electroless Nickel Immersion Gold)**

O acabamento ENIG é composto por uma camada de níquel químico coberta por uma fina camada de ouro, depositada por imersão. Essa combinação oferece excelente planicidade e alta confiabilidade para componentes sensíveis e densos, como BGAs e circuitos SMD avançados.

**Características Técnicas**:

* Superfície plana, ideal para componentes de alta densidade.
* Maior resistência à oxidação e ao desgaste.
* Compatibilidade com processos de soldagem de alta precisão.
* Conformidade com regulamentações ambientais (Lead-Free).
* Custo: Alto.

**Recomendações**: Preferido em projetos de alta complexidade e precisão, como eletrônicos industriais, dispositivos médicos e equipamentos de telecomunicações.

#### **Escolha do Acabamento para o Projeto VoltBridgeDAC**

Para o protótipo **VoltBridgeDAC**, optou-se pelo acabamento **HASL** devido à sua relação custo-benefício e à adequação às necessidades de prototipagem e testes iniciais. O HASL oferece uma proteção confiável contra oxidação e facilita a soldagem dos componentes em uma produção inicial.

No entanto, para projetos futuros ou versões mais avançadas do VoltBridgeDAC, recomenda-se considerar o acabamento **ENIG**, principalmente se o projeto envolver maior densidade de componentes SMD, BGAs ou requisitos rigorosos de qualidade e confiabilidade.

A escolha do acabamento ideal para uma PCB deve sempre equilibrar as exigências técnicas, a durabilidade esperada e os custos de produção, garantindo a entrega de um produto de alta qualidade e performance.

## **NÚMERO DE CAMADAS DA PCB**

A escolha do número de camadas em uma PCB (Placa de Circuito Impresso) é um fator crucial no processo de design, diretamente influenciado pelas exigências do projeto, como complexidade do circuito, densidade de componentes e custo de fabricação.

#### **CLASSIFICAÇÃO DE CAMADAS**

* **Placas de Camada Simples (Single-Layer)**: Contêm apenas uma camada condutiva, ideal para projetos simples e de baixo custo.
* **Placas de Dupla Camada (Double-Layer)**: Apresentam duas camadas condutivas, permitindo maior densidade de componentes e maior flexibilidade no roteamento.
* **Placas Multicamadas (Multilayer)**: Com três ou mais camadas condutivas, são projetadas para aplicações complexas e circuitos de alta densidade.

#### **FATORES IMPORTANTES NA ESCOLHA DA QTD. DE CAMADAS**

* **Complexidade do Circuito**: Projetos com maior número de conexões ou circuitos mais densos geralmente exigem mais camadas para acomodar o roteamento e evitar cruzamentos de trilhas.
* **Custo**: Placas de camada única são mais econômicas, enquanto as multicamadas têm custos mais elevados devido aos processos de fabricação mais complexos.
* **Tamanho da PCB**: Para designs compactos, o uso de mais camadas pode ser necessário para otimizar o espaço.
* **Desempenho Elétrico**: PCBs multicamadas oferecem melhor desempenho em frequências altas, maior imunidade a interferências eletromagnéticas (EMI) e maior integridade de sinal.

#### **Escolha para o Projeto VoltBridgeDAC**

O projeto **VoltBridgeDAC** foi concebido para ser produzido em uma PCB de **camada única** (Single-Layer). Essa decisão baseia-se em:

* A **simplicidade do circuito**, que não requer trilhas cruzadas complexas nem uma alta densidade de componentes.
* O **controle de custos**, essencial para a fase de prototipagem e para manter a acessibilidade do dispositivo.
* A **viabilidade técnica**, visto que os requisitos elétricos e mecânicos do projeto podem ser atendidos com eficiência por uma única camada condutiva.

Embora essa escolha seja ideal para o estágio atual, futuros aprimoramentos ou versões mais avançadas do VoltBridgeDAC podem explorar designs com camadas adicionais, caso o projeto exija maior densidade de componentes, melhores propriedades de blindagem ou maior miniaturização.

A definição do número de camadas reflete um compromisso entre os requisitos do circuito, os custos de produção e o desempenho esperado, garantindo que o design atenda às necessidades específicas do projeto de forma eficiente.