DISCIPLINA EA701
Introdução aos Sistemas Embarcados

ROTEIRO 10: Sensores e Atuadores
Transdutores, Sensores e Atuadores Programáveis, Interfaces de Comunicação Serial Síncrona SPI e I2C, EEPROM 24LC1, MPU-6065, *Display* Nokia

Profs. Antonio A. F. Quevedo e Wu Shin-Ting

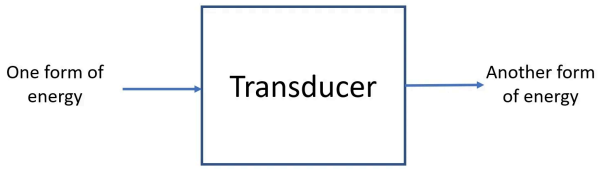
FEEC / UNICAMP

Revisado em outubro de 2024

INTRODUÇÃO

Após utilizarmos diversos sensores, como o botão azul, o *encoder* e o *keypad* de membrana, e atuadores digitais, como LEDs e um motor DC de 5V, nos roteiros anteriores, e explorarmos sensores analógicos, como o potenciômetro, sensor de temperatura e *joystick*, assim como atuadores analógicos, como *buzzers*, no Roteiro 9, este Roteiro se concentra em uma categoria específica de dispositivos: sensores e atuadores digitais com interfaces de comunicação serial síncrona. A principal diferença entre esses dispositivos e os discutidos anteriormente está na forma como se conectam e se comunicam com o microcontrolador. Enquanto os dispositivos digitais e analógicos podem usar diversas interfaces, os sensores e atuadores com interfaces de comunicação serial se beneficiam da simplicidade e eficiência das comunicações seriais síncronas. Essas comunicações diferem das assíncronas, abordadas no Roteiro 8, pois, nas síncronas, receptor e transmissor compartilham um sinal de relógio, permitindo que a

transmissão de dados ocorra de forma coordenada sem os *bits* de sincronização. Isso resulta em um desempenho superior na transmissão de dados.



Transdutores são dispositivos que convertem um tipo de energia em outro tipo. Sensores e atuadores são classes específicas de transdutores que realizam a conversão entre energia elétrica e outras formas de energia, como mecânica (motor), luminosa (lâmpada) e sonora (microfone). Essa conversão é fundamental porque permite que a energia elétrica seja transformada em tensões que podem ser facilmente processadas pelos módulos periféricos de um microcontrolador. Além disso, essa capacidade de conversão facilita a integração do microcontrolador em diversos ambientes físicos, permitindo-lhe interagir com o mundo ao seu redor. Ele pode receber informações do ambiente por meio de sensores e executar ações através de atuadores, tudo de forma eficiente e transparente. Essa interação abre a porta para a implementação de sistemas automatizados e interativos em uma ampla variedade de aplicações.

A predominância de sensores e atuadores com interfaces de comunicação serial síncrona, como I2C, SPI e CAN, pode ser atribuída a diversos fatores. Primeiro, a necessidade de reduzir o número de pinos necessários para a comunicação é um aspecto fundamental em projetos de eletrônica, onde o espaço é frequentemente limitado. As interfaces de comunicação serial síncrona permitem a interconexão de múltiplos dispositivos usando um número reduzido de fios, facilitando a construção de sistemas mais compactos e organizados. Além disso, a robustez e a flexibilidade dos protocolos de comunicação serial síncrona tornam-nos ideais para aplicações que exigem comunicação rápida e confiável. Em ambientes onde múltiplos dispositivos precisam compartilhar informações, a capacidade de priorizar a comunicação em série se torna um ativo valioso.

Devido às características distintas dos sensores e atuadores, um planejamento cuidadoso é fundamental em projetos de sistemas embarcados, tanto para a aquisição dos sinais gerados pelos sensores quanto para a geração dos sinais de controle adequados para os atuadores. Nesse contexto, antes de explorarmos os módulos integrados nos microcontroladores — que desempenham um papel crucial na interligação desses componentes com os processadores — dedicamos uma seção deste Roteiro a uma apresentação abrangente dos diversos sensores e atuadores, assim como de suas tecnologias, essenciais para a escolha adequada que atenda a necessidades específicas. Além disso, examinaremos em detalhe dois protocolos de comunicação serial síncrona, I2C e SPI, amplamente suportados pela maioria dos microcontroladores.

Neste roteiro, vamos explorar três projetos que demonstram como configurar e controlar sensores e atuadores modernos utilizando microcontroladores. No primeiro projeto, vocês conhecerão um atuador programável: o *display* Nokia, que transforma sinais elétricos em imagens digitais, tudo isso com a ajuda do módulo SPI do microcontrolador STM32H7A. Nos projetos seguintes, exploraremos a integração de dois periféricos via I2C: uma memória EEPROM que guarda dados de forma eficiente e o sensor de movimentos MPU-6050, que capta movimentos em 6 eixos. Ao longo dessa experiência, vocês descobrirão como as bibliotecas de funções compartilhadas pelos desenvolvedores facilitam a configuração dos periféricos, tornando a programação desses dispositivos mais simples.

PROJETOS-EXEMPLO

Nesta seção, exploraremos dois projetos fascinantes que demonstram o potencial da comunicação serial entre sensores, atuadores proramáveis e microcontroladores em aplicações práticas. O primeiro projeto, voltado para a saída, permite a visualização de informações em tempo real, enquanto o segundo, de entrada, utiliza um acelerômetro para captar informações do ambiente, Antes de nos aprofundarmos nos detalhes da programação do acelerômetro, apresentaremos um projeto específico de aplicação de uma memória EEPROM, ilustrando como programar uma interface de comunicação serial síncrona I2C.

Projeto de comunicação com um *display* SPI

Os *displays* LCD de texto, como o nome diz, recebem códigos ASCII e apresentam diretamente os caracteres correspondentes. Em *displays* gráficos, a informação transferida ao dispositivo se refere ao estado de cada *pixel*. Assim, a quantidade de dados transmitida é muito maior que para um *display* de texto, exigindo interfaces com boa velocidade de comunicação. O *display* Nokia 5110 possui 84x48 *pixels* monocromáticos (um *bit* por *pixel*), ou seja, um total de 4.032 *bits* por quadro. Por isso, ele utiliza a interface SPI, em modo unidirecional, funcionando como *slave* de um microcontrolador. Note que para escrever texto neste tipo de *display*, é necessário “desenhar” os caracteres, definindo os *pixels* a serem ativados para cada um.

A questão que surge é como realizar esses desenhos em tempo real de maneira diversificada. Uma abordagem é enviar uma série de instruções apropriadas para os registradores do *display* Nokia, programando-o através do microcontrolador. Essas instruções são convertidas em sinais de controle que atuam diretamente sobre o *display*. Vamos demonstrar isso, implementando um projeto com o *display* Nokia5110, usando uma **biblioteca** para simplificar a implementação. A **biblioteca** nada mais é que um par de arquivos adicionados ao projeto, sendo um do tipo *header* (extensão “.h”, com macros e protótipos das funções) e um do tipo código-fonte C (extensão “.c”, com a implementação das funções).

1. Crie um projeto chamado “SPI_Nokia”, sem inicializar os periféricos. Ative o *Debug* e mantenha o *clock* geral em 64MHz (configurações padrão). Gere o código.

2. Descompacte o arquivo “zip” fornecido para o roteiro, em uma pasta temporária. Dos vários arquivos disponíveis, vamos usar dois neste projeto: “Nokia_5110.c” e “Nokia_5110.h”. Precisamos incluir estes arquivos dentro do projeto. Eles podem ser incluídos em qualquer lugar dentro da pasta “Core”, mas aqui, para manter a organização dos arquivos, vamos colocar o “Nokia_5110.c” na pasta “Src” e o “Nokia_5110.h” na pasta “Inc”. Para isso, há três formas possíveis. A primeira é criar arquivos vazios dentro dss pastas clicando com o botão **direito** na pasta e escolhendo a opção “New - Source File” para o “.c” e “New - Header File” para o “.h”. Depois é necessário copiar o texto de cada arquivo original e colar no arquivo novo correspondente.

As outras duas formas são mais diretas. Em uma delas, abre-se a pasta onde se deseja colocar o arquivo, clicando na mesma com o botão direito, escolhendo “Properties”, e na linha “Location” clicar no botão no limite da linha à direita (*Show in System Explorer*). Em seguida cada arquivo da biblioteca é movido para a pasta correspondente. Por fim, seleciona-se no *Project Explorer* do IDE a raiz do arquivo, clica-se com o botão direito e seleciona-se a opção “Refresh” (ou usa-se a tecla F5). Assim, os arquivos colocados nas pastas serão listados no projeto.

A última forma é mais simples ainda. No Explorador de Arquivos, clica-se no arquivo e arrasta-se o mesmo diretamente para a pasta no *Project Explorer* do IDE. Neste caso, abre-se uma caixa de diálogo. Escolha a opção “Copy files” para que seja feita umc cópia completa do arquivo dentro do projeto, e não apenas um *link* para o arquivo original.

3. Com as bibliotecas incluídas, os arquivos podem ser abertos e editados. Abra os arquivos e veja a estrutura dos mesmos. Note que cada implementação de função apresenta um cabeçalho de comentários com explicações sobre o uso de cada uma. A sintaxe deste cabeçalho pode parecer estranha, mas ela é usada para permitir um aplicativo chamado **Doxygen** gerar a documentação de bibliotecas automaticamente, a partir dos comentários das funções. Mesmo sem o Doxygen, os comentários são facilmente compreensíveis pelos programadores.

4. Abra o arquivo “main.c”. Precisamos inicialmente incluir a biblioteca na compilação. Para isso, na seção `/* USER CODE BEGIN Includes */`, adicione as linhas:

```
#include "Nokia_5110.h"
```

Além disso, vamos definir macros para controlar o pino GPIO ligado ao controle da *backlight*. Na seção `/* USER CODE BEGIN PD */`, adicione as macros:

```
// Macros para backlight
#define BKLT1()    GPIOD->BSRR = GPIO_BSRR_BS5
#define BKLT0()    GPIOD->BSRR = GPIO_BSRR_BR5
```

Vamos também adicionar um vetor com o *bitmap* do logo da FEEC. Este vetor possui a sequência de *bytes* a serem transmitidos ao *display* para que desenhem o *bitmap*. Para isso, o logo foi submetido a um app *online* que gera a sequência de *bytes* no formato adequado. Este

vetor deve residir na FLASH, e para isso ele é declarado como “static const”. Na seção /* USER CODE BEGIN PV */, declare o vetor:

```
static const char LogoFEEC[] = {
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x18, 0x5c, 0xd8, 0x07, 0x1e, 0xe8, 0xf0, 0xc0, 0x80, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0, 0xe0, 0x70,
0xb0, 0xc0, 0xc0, 0x00, 0x02, 0x0f, 0x1e, 0x39, 0xe7, 0xcf, 0x3e, 0x78, 0xf0,
0xc0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfc, 0xff, 0xff, 0x1f, 0x1f, 0x1f,
0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x80, 0xf0, 0xf8, 0xfe, 0x3e, 0x3f, 0x1f, 0x1f,
0x3e, 0xfe, 0xf8, 0xf0, 0x80, 0x00, 0x00, 0xc0, 0xf0, 0xf8, 0x7e, 0x3e, 0x1f,
0x1f, 0x3f, 0x3e, 0xfe, 0xf8, 0xf0, 0x00, 0x00, 0x00, 0xc0, 0xf0, 0xfc, 0x7e,
0x3f, 0x3f, 0x3f, 0x3f, 0x3e, 0x7e, 0x18,
        0x00, 0x00, 0x00, 0x80, 0xc0, 0x10, 0xe0, 0xbe, 0xde, 0x13, 0x7d, 0x1e,
0x0f, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xe0, 0xf1, 0xff, 0xfe, 0xfd,
0x73, 0x0f, 0x1f, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xf8, 0xf8, 0xf8,
0xf8, 0x78, 0x38, 0x00, 0x00, 0xff, 0xff, 0xff, 0x7c, 0x7c, 0x7c, 0x7c, 0x7c,
0x7c, 0x7c, 0x7f, 0x3f, 0x0f, 0x00, 0x78, 0xff, 0xff, 0xff, 0x7c, 0x7c, 0x7c,
0x7c, 0x7c, 0x7c, 0x7c, 0x7f, 0x3f, 0x0f, 0x00, 0xff, 0xff, 0xff, 0x03, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x70, 0x7c, 0xbe, 0x87, 0x00, 0xc9, 0xde, 0x07, 0x03, 0x00, 0x00, 0x00,
0x00, 0x00, 0x80, 0xb0, 0x38, 0x7e, 0x7f, 0x7f, 0x7f, 0x3f, 0x0f, 0x07, 0x01,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x3f, 0x7f, 0xfc, 0xf8, 0xf0, 0xf0, 0xf0,
0xf0, 0xf8, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x07, 0x3f, 0x7f, 0xfc, 0xf0, 0xf0,
0xf0, 0xf0, 0xf0, 0xf8, 0xf0, 0x00, 0x00, 0x00, 0x01, 0x0f, 0x3f, 0xff, 0xfc,
0xf0, 0xf0, 0xf0, 0xf0, 0xf8, 0xfc, 0x60,
        0x00, 0x01, 0x03, 0x07, 0x03, 0x7e, 0xf9, 0xcf, 0x3e, 0xf8, 0xf0, 0xe0,
0x80, 0x03, 0x0f, 0x0f, 0x00, 0x07, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x03, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x03, 0x01,
0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01,
0x03, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
0x01, 0x01, 0x03, 0x01, 0x01, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x07, 0x1d, 0x33, 0x4f,
0x7f, 0x7e, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
```

5. Agora vamos inicializar o *display* e desenhar o logo da FEEC. Na seção /* USER CODE BEGIN 1 */ declare a variável com inicialização

```
uint8_t contraste = 52;
```

Na seção `/* USER CODE BEGIN 2 */`, faça a chamada das funções da biblioteca:

```
lcdBegin();
setContrast(contraste);
clearDisplay(WHITE);
SetBitMap(LogoFEEC);
updateDisplay();
```

Na primeira linha, estamos inicializando a interface SPI e enviando a configuração inicial para o *display*. Na segunda linha, o contraste é ajustado. Na terceira linha, o *buffer* do *display* é limpo e na quarta linha ele é preenchido com o *bitmap*. A quinta linha atualiza o *display*. Ou seja, todos os comandos da biblioteca alteram os *bits* de um *buffer* na RAM do microcontrolador. Após todos os comandos desejados serem executados, deve-se chamar a função "updateDisplay" para que os *bits* dos *pixels* sejam enviados do *buffer* para o *display*.

Note que o contraste varia entre módulos. O valor indicado costuma funcionar bem para a maioria dos módulos. Se a imagem aparecer muito "fraca", deve-se aumentar o contraste. Ao contrário, se a imagem aparecer com as áreas claras muito escurecidas, deve-se diminuir o contraste.

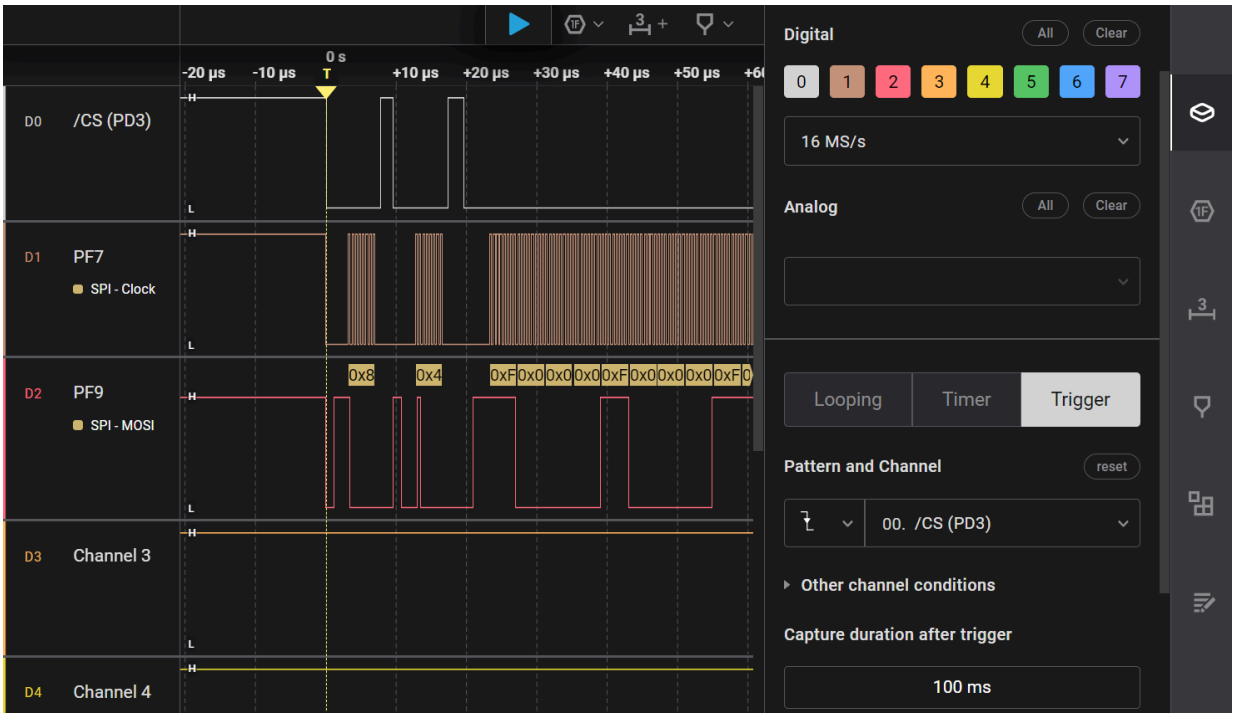
6. Para fins de demonstração, vamos usar, além da carga do *bitmap*, o traçado de linhas, o uso de fundo preto e linhas brancas, e a apresentação de texto. Após a linha `/* USER CODE BEGIN 3 */`, escreva o código do *loop*:

```
HAL_Delay(2000);
clearDisplay(BLACK);
// Desenha cubo com linhas claras em fundo escuro
setRect(25, 10, 45, 30, 0, WHITE);
setRect(35, 20, 55, 40, 0, WHITE);
setLine(25, 10, 35, 20, WHITE);
setLine(45, 30, 55, 40, WHITE);
setLine(25, 30, 35, 40, WHITE);
setLine(45, 10, 55, 20, WHITE);
updateDisplay();
BKLT1();
HAL_Delay(2000);
clearDisplay(WHITE);
// Escreve texto escuro em fundo claro
setStr("EA701 FEEC", 0, 0, BLACK);
setStr("Profs.", 0, 8, BLACK);
setStr("Quevedo e Ting", 0, 16, BLACK);
updateDisplay();
HAL_Delay(2000);
clearDisplay(WHITE);
SetBitMap(LogoFEEC);
updateDisplay();
HAL_Delay(2000);
clearDisplay(WHITE);
updateDisplay();
BKLT0();
```

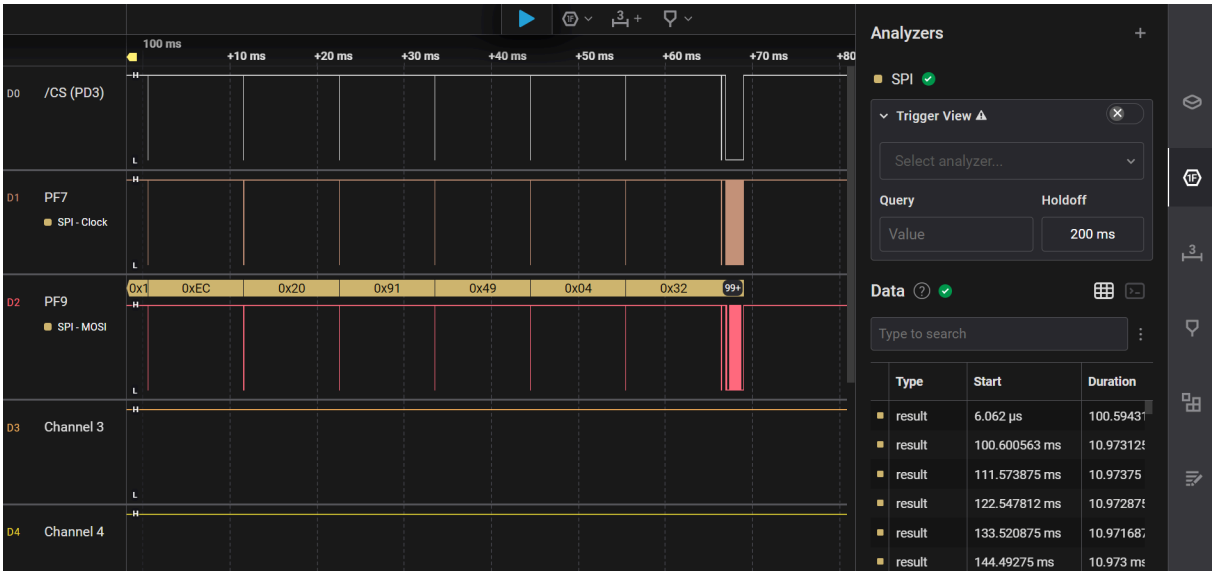
7. Realize o *Build* e transfira o código executável para o microcontrolador no modo *Debug*. Execute o programa e veja a demonstração no *display*.

8. Coloque um breakpoint na linha "`setContrast(contraste);`" da função `main`. Em seguida, reinicie o programa ("Terminate and Relaunch"). Quando a execução pausar nessa linha, altere o valor da variável `contraste` na aba "Variables", aumentando seu valor. Depois, continue a execução ("Resume"). Repita o procedimento, desta vez reduzindo o valor da variável. Registre os efeitos observados em diferentes situações e configure o nível de contraste de sua preferência.

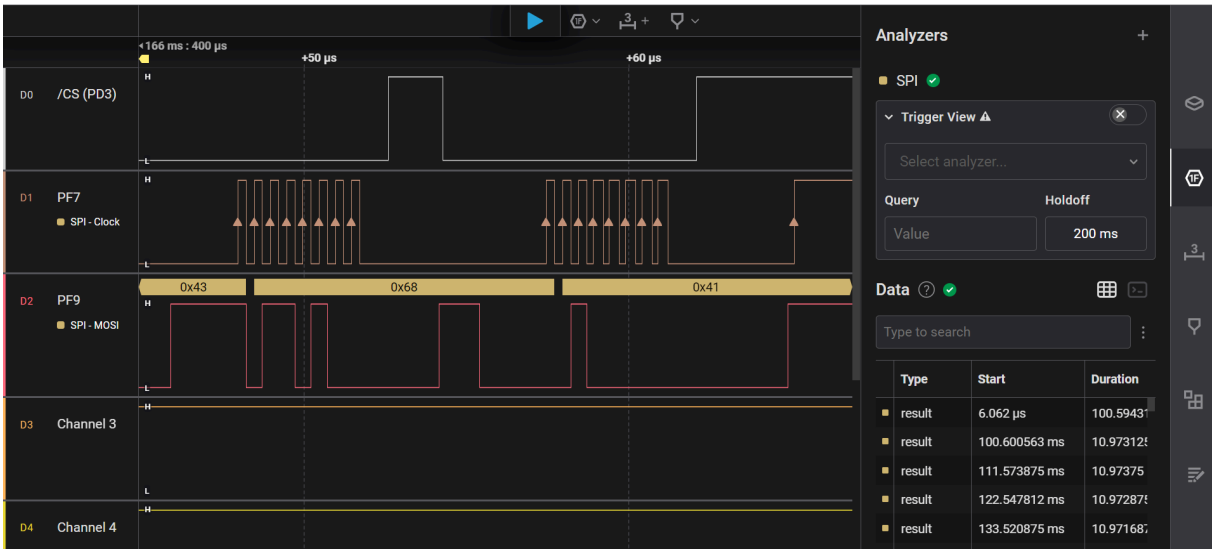
8. Remova o *breakpoint*. Ligue os 3 primeiros canais do analisador lógico nos pinos adequados do conector "Zio" (parte superior da placa NUCLEO). Veja a figura 18, página 37, do manual da placa NUCLEO para localizar os sinais corretos: canal 0 em PD3 (/CS), canal 1 em PF7 (SCK) e canal 2 em PF9 (MOSI). Configure a aquisição do "Logic" para um *trigger* na borda de descida do canal 0 (/CS) e um tempo de aquisição de 200ms.



9. Coloque um *breakpoint* na linha "LCDcommand(0x21);" da função "lcdBegin" do arquivo "Nokia_5110.c". Reinicie ("Terminate and Relauch") a execução até o *breakpoint*, inicie a aquisição no "Logic" e continue a execução do programa. Adicione o analisador SPI nos sinais, mantendo as configurações padrão e ajustando os canais para cada sinal. Veja a sequência de *bytes* de comando enviados e compare com o *datasheet do display* (Capítulo 8 - INSTRUCTIONS, e em especial a tabela 1 na página 14).



10. Faça um "zoom in" na vizinhança de um dos "palitos". Interprete a relação entre os sinais visualizados, identificando o instante em que o dispositivo escravo (*display*) é selecionado e o instante em que os quadros de dados são enviados para o *display*.



Projeto de comunicação com uma EEPROM I2C

As EEPROMs com interface I2C são muito úteis quando se necessita armazenar uma quantidade pequena de dados de forma semi-permanente. Uma aplicação comum é no armazenamento de parâmetros de configuração do sistema, que podem ser alterados mas não mudam com frequência, e precisam permanecer armazenados mesmo quando o sistema se encontra desligado. Embora não se classifique como um sensor ou atuador — já que apenas dados são transferidos entre a memória EEPROM e o microcontrolador, sem resposta ativa — decidimos incluí-la para ilustrar a programação do módulo I2C, permitindo a leitura e escrita

de dados em um CI EEPROM 24LC16 através do protocolo I2C. Sendo 24C16 o modelo de EEPROM implementado na placa de Expansão, a capacidade de armazenamento dessa memória é 16 kilo*bits* (ou 2 kilo*bytes*), podendo ser escrita em blocos de no máximo 16 *bytes*. O *device address* de EEPROM é 0b1010xxx que deve ser alinhado à esquerda em conformidade com o protocolo I2C.

1. Crie um projeto chamado "I2C_EEPROM", sem inicializar os periféricos. Ative o *Debug* e mantenha o *clock* geral em 64MHz (configurações padrão). Gere o código.

2. Adicione os arquivos "EEPROM_I2C.c" e "EEPROM_I2C.h" nas pastas "Src" e "Inc", como foi feito no projeto anterior.

3. Abra o arquivo "main.c". Vamos incluir a biblioteca no projeto. Na seção `/* USER CODE BEGIN Includes */`, adicione a inclusão:

```
#include "EEPROM_I2C.h"
```

4. Vamos criar variáveis locais para a programação. Na seção `/* USER CODE BEGIN 1 */`, declare as variáveis para os *buffers* de dados, para os *loops* e para receber os códigos de erro:

```
uint8_t bw[16], br1[32], br2[16]; // Buffers de escrita e de leitura
uint8_t k; // Auxiliar de loops
uint8_t errno; // Variavel de erro
```

5. O programa de demonstração será executado apenas uma vez, antes do *loop* infinito. Vamos escrever e ler diferentes conjuntos de dados para demonstrar o funcionamento da EEPROM. Na seção `/* USER CODE BEGIN 2 */`, escreva o código:

```
I2C_Init();
err = 0;
// Preenche os buffers de leitura com 0xFF
for(k = 0; k < 32; k++) {
	br1[k] = 0xFF;
}
for(k = 0; k < 16; k++) {
	br2[k] = 0xFF;
}
for(k = 0; k < 16; k++) {
	bw[k] = 0; // Escreve uma pagina com zeros
}
errno = Mem_Write(32, bw, 16); // Posicao de memoria 32-47 (pagina 2)
HAL_Delay(200); // Atraso para garantir o fim da escrita de bloco
for(k = 0; k < 16; k++) {
	bw[k] = k; // Escreve sequencia de bytes 0-15
}
errno = Mem_Write(0, bw, 16); // Posicao de memoria 0-15 (pagina 0)
HAL_Delay(200);
```

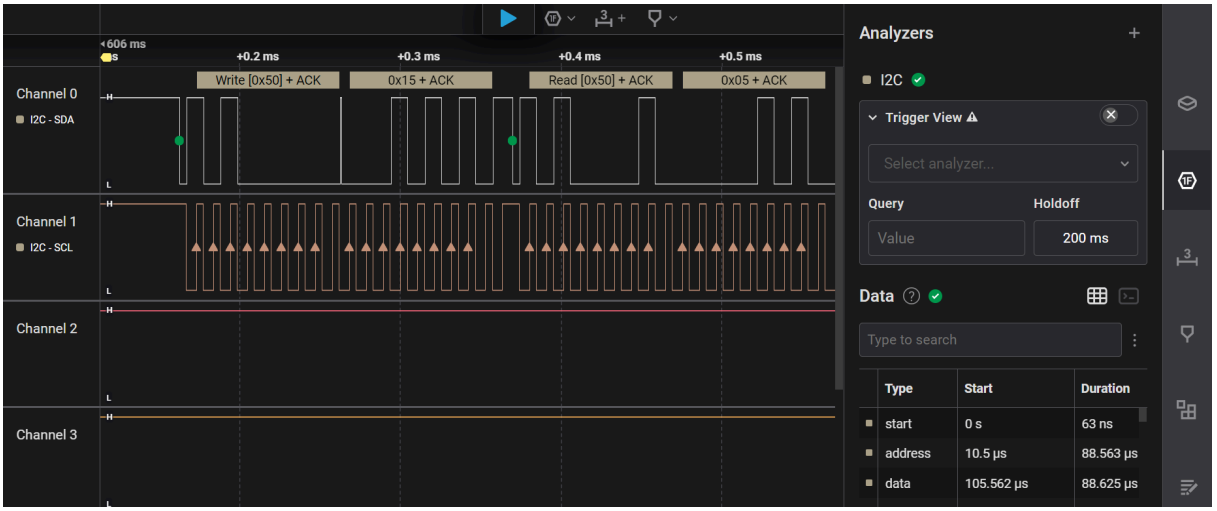
```
for(k = 0; k < 16; k++) {
      bw[k] = k + 16; // Escreve sequencia de bytes 16-31
}
errno = Mem_Write(16, bw, 16); // Posicao de memoria 16-31 (pagina 1)
HAL_Delay(200);
errno = Mem_Read(0, br1, 32); // Le paginas 0 e 1 em uma operacao

errno = Mem_Read(32, br2, 16); // Le pagina 2
HAL_Delay(10); // Apenas para poder colocar um breakpoint
```

Os três argumentos das funções `Mem_Write` e `Mem_Read` correpondem, respectivamente, ao endereço da memória 24LC16, que vai de 0 a (2*1024)-1, o endereço do vetor de *bytes* a serem escritos ou lidos e a quantidade de *bytes* a serem escritos ou lidos. As duas funções retornam com "1" em caso de erro e "0" se a execução for concluída com sucesso. Foram realizados três acessos de escrita em blocos de 16 *bytes*. Primeiro, no intervalo de endereços de 32 a 47, foram escritos valores 0x00. Em seguida, no intervalo de 0 a 15, foram gravados os caracteres de 'A' a 'P'. Por último, no intervalo de 16 a 31, foram armazenados os valores de 0 a 15. Após essas operações, foram feitos dois acessos aleatórios para leitura de um bloco de 32 *bytes* a partir do endereço 0 e de 16 *bytes* a partir do endereço 32. Sendo uma memória de tamanho de 16 k*bits* = 2k*bytes*, é possível gravar até 2x1024 *bytes* neste memória. Note que toda operação de escrita só pode envolver um bloco de memória com alinhamento de 16 *bytes*, enquanto que operações de leitura podem passar os limites de blocos.
O preenchimento dos *buffers* de leitura com o valor 0xFF é para garantir que após a execução do programa os mesmos foram realmente preenchidos com os valores lidos na EEPROM. O valor 0xFF não é usado em nenhum *byte* gravado na EEPROM, pelo menos neste projeto.

6. Coloque um *breakpoint* na linha "HAL_Delay(10);". Faça o *Build* e transfira o executável para o microcontrolador no modo *Debug*. Execute o código e observe o conteúdo dos vetores "br1" e "br2".

7. Conecte os canais 0 e 1 do analisador lógico aos pinos PB8 (SCL) e PB9 (SDA). Configure a aquisição para 2s e o *trigger* para borda de descida no canal do SDA. Execute novamente ("Terminate and Relaunch") o programa e faça a coleta ao chamar uma função de escrita e também uma de leitura. Configure o analisador para o modo I2C e compare os sinais lidos com o esperado para cada caso (leitura e escrita).

8. Note o uso dos 3 *bits* menos significativos do *device address* e do primeiro *byte* transmitido para definir o endereço de memória dentro da EEPROM a ser acessado, bem como o uso de uma escrita do endereço seguida de um "REPEATED START" para definir o endereço antes de uma operação de leitura. Com base na captura de um acesso de leitura abaixo, qual é o *device address* e qual é o endereço inicial de acesso na memória? Como a condição "REPEATED START" é sinalizada no diagrama de tempo?



Projeto de comunicação com um sensor de movimento I2C

Uma memória EEPROM apenas armazena *bytes* em endereços diversos. Um sensor "inteligente" usa a interface digital para receber parâmetros de configuração e enviar os dados coletados, já na forma digital. Isto é feito através de **registradores** internos ao sensor, sendo que cada um ocupa um endereço. A semelhança entre os acessos dos registradores do sensor e um acesso à memória EEPROM via I2C é como a semelhança entre um acesso à memória RAM e a um registrador de periférico interno em uma MCU. No caso do MPU6050, um sensor de movimentos em 6 eixos, os endereços de registradores são de 1 *byte*, e assim não são usados *bits* do *device address* para completar o endereço.

1. Crie um projeto chamado "I2C_MPU6050", sem inicializar os periféricos. Ative o *Debug* e mantenha o *clock* geral em 64MHz (configurações padrão). Gere o código.

2. Adicione os arquivos "MPU6050.c" e "MPU6050.h" nas pastas "Src" e "Inc", como foi feito nos projetos anteriores.

3. Abra o arquivo "main.c". Vamos incluir a biblioteca no projeto. Na seção `/* USER CODE BEGIN Includes */`, adicione a inclusão:

```
#include "MPU6050.h"
```

4. Vamos criar variáveis para a programação. Na seção `/* USER CODE BEGIN 1 */`, declare as variáveis para funções auxiliares:

```
uint8_t r, id;
```

Na seção `/* USER CODE BEGIN PV */`, crie as variáveis para guardar os valores dos eixos de aceleração e velocidade angular. Estas variáveis devem ser **globais** para que o recurso a ser usado neste projeto possa funcionar.

```
int16_t ax, ay, az, gx, gy, gz;
```

5. Agora vamos configurar a interface I2C e o sensor MPU. Na seção `/* USER CODE BEGIN 2 */`, faça a chamada das funções:

```
I2C_Init();
id = MPU_Init();
HAL_Delay(10); // Apenas para poder adicionar um breakpoint
```

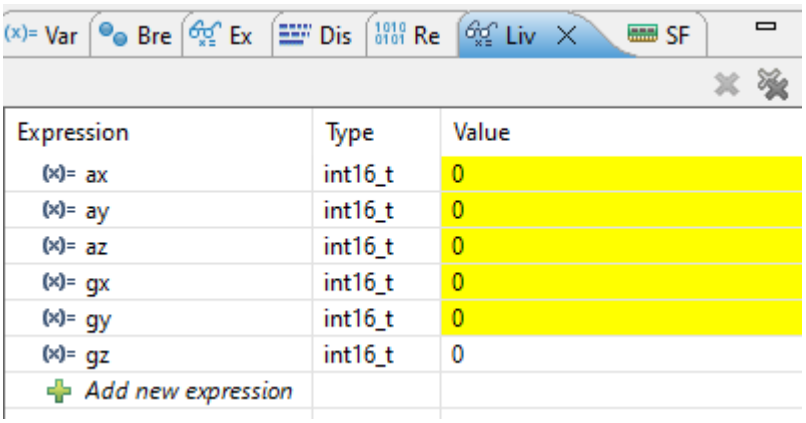
Note que a variável "id" recebe o *byte* de identificação do sensor, que deve ser 0x68.

6. Agora vamos adicionar o código executado em *loop*. Abaixo da linha `/* USER CODE BEGIN 3 */`, escreva o código:

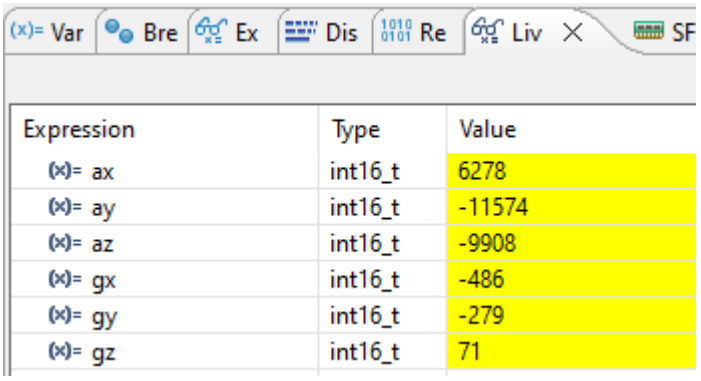
```
// Espera o "Data ready" (Bit 0 de INT_STATUS em 1)
r = 0x00;
while(!r) {
	Reg_Read(INT_STATUS, &r);
	r &= 0x01;
}
// Lê os valores de aceleração e de velocidade angular
GetAccel(&ax, &ay, &az);
GetGyro(&gx, &gy, &gz);
```

7. Faça o *Build*. Conecte o módulo do MPU6050 no *header* H4 (I2C1). Há uma marcação indicando o pino 1, que é o mais à esquerda no *header* (alinhado com o pino 1 do *header*) Este é o pino onde nenhum fio está conectado. Adicione um *breakpoint* na linha "HAL_Delay(10);". Transfira o código executável para o microcontrolador no modo *Debug*, e inicie ("Resume") o programa. Ao pausar, observe o valor da variável "id", que deve conter o identificador do sensor retornado pela função MPU_Init().

8. Agora vamos usar um recurso do Eclipse denominado *Live expressions*. Estas são como as *Expressions* comuns no modo *debug*, porém são atualizadas em tempo real. É importante ressaltar que este recurso funciona apenas com variáveis globais. Vá na aba *Live Expressions* e adicione as variáveis ax, ay, ax, gx, gy e gz. Ajuste as larguras das colunas para que se possa ver os valores das variáveis.



9. Remova o *breakpoint* do item 7 e faça o *Resume*. Veja as variáveis sendo atualizadas continuamente. Movimente o sensor e veja os valores de “a” e “g” mudarem.



10. Os valores amostrados pelo acelerômetro são representados em complemento de 2, utilizando 16 *bits*, com uma faixa de medição de [-2g, 2g]. Considerando que a relação entre os valores representados e a faixa de valores medidos é linear, quais são os fatores de g correspondentes aos valores mostrados em ax, ay e az no item 9?

11. Conecte o analisador lógico como no item 7 da EEPROM e visualize as leituras periódicas dos valores amostrados.

SENSORES E ATUADORES

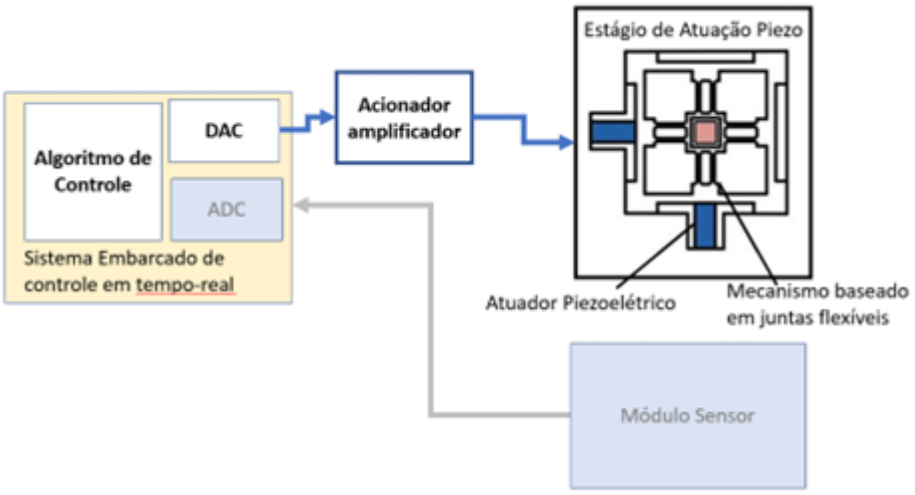
Os **sensores** são capazes de medir uma ampla gama de variáveis, como temperatura, pressão, luz, som e muito mais. Eles traduzem o mundo físico em dados digitais, fornecendo informações para que os microcontroladores tomem decisões. Em essência, os sensores são os órgãos dos sentidos de um sistema microcontrolado, transmitindo constantemente informações para o “cérebro” do sistema. Por outro lado, os **atuadores** são os músculos das máquinas, capazes de executar ações com base nas decisões programadas nos “cérebros” dos microcontroladores. Eles podem girar um motor, acionar uma válvula, mover um braço de robô ou alterar a intensidade da luz emitida por um LED, como realizar outras tarefas. Os atuadores transformam as decisões tomadas por um sistema microcontrolado em ações, tornando as máquinas capazes de interagir com o ambiente.

Em sistemas eletrônicos, os sensores normalmente geram uma tensão que é proporcional à quantidade física a ser medida. Essa tensão pode ser convertida em uma representação numérica digital por meio de um conversor analógico-digital (ADC), conforme discutido no Roteiro 9. Sensores que possuem um ADC integrado são chamados de **sensores digitais**, enquanto aqueles que não o possuem são classificados como **sensores analógicos**. Por outro lado, os atuadores geralmente são acionados por uma tensão que pode ser gerada por um conversor digital-analógico (DAC), também abordado no Roteiro 9. Atuadores com DAC integrado são classificados como **atuadores digitais**, enquanto **atuadores analógicos** recebem diretamente os sinais analógicos, que consistem em uma tensão ou corrente que varia

continuamente dentro de um intervalo específico. Mais recentemente, estão disponíveis no mercado alguns *chips* que integram sensores digitais ou atuadores digitais junto com microprocessadores.

Com a evolução da tecnologia de materiais, tornou-se viável medir uma ampla gama de parâmetros, como aceleração, pressão, temperatura e rotação, além de alterar a posição de componentes ou gerar vibrações por meio de tecnologias microeletromecânicas. Essa inovação permitiu o desenvolvimento de dispositivos de sensoreamento e atuadores microscópicos, conhecidos como **sistemas micro-eletromecânicos** (em inglês, *Micro-Eletromecanical Systems* – MEMS). Devido ao seu tamanho compacto, baixo consumo de energia e precisão em tempo real, é possível criar **sensores** e **atuadores inteligentes** ao combinar MEMS com eletrônica adicional, como microcontroladores ou sistemas de processamento de sinais. Quando esses sensores e atuadores inteligentes são integrados a interfaces de rede, eles podem formar redes de sensores e atuadores que, conectadas à *internet*, possibilitam o acesso remoto a esses dispositivos.

A figura a seguir ilustra um cenário típico de um sistema embarcado. Neste contexto, o microcontrolador interpreta os sinais recebidos de um sensor e, em resposta, gera os sinais de controle necessários para acionar o atuador. Devido à ampla variação nos níveis de tensão e potência dos atuadores, que são essenciais para realizar diversas atividades no mundo físico analógico, o uso de circuitos acionadores, também conhecidos como *drive*, é fundamental no projeto de sistemas embarcados. Dessa forma, cabe aos projetistas selecionar criteriosamente os componentes eletrônicos complementares adequados para criar uma interface eficaz entre o mundo digital e o físico. Durante esse processo, a escolha de sensores e atuadores deve ser feita com atenção, levando em consideração não apenas a qualidade, o desempenho e o custo, mas também a complexidade dos projetos de interface complementares.



A classificação de sensores e atuadores fornece aos desenvolvedores uma abordagem estruturada para compreender as opções disponíveis, incluindo as tecnologias, faixas de operação, resposta dinâmica e complexidade de circuitos de interface com microcontroladores. Essa organização sistemática é crucial em um mercado com tantas

opções, pois cada categoria de sensor e atuador apresenta uma ampla variação de preços, bem como vantagens e desvantagens particulares. Uma taxonomia clara facilita a escolha dos dispositivos mais adequados com base nas necessidades específicas de cada projeto e ajuda a reduzir o tempo de desenvolvimento e minimiza erros de seleção, resultando em soluções mais eficientes e eficazes para um projeto atenda aos objetivos desejados dentro do orçamento e das restrições técnicas.

Classificação de sensores

Cada sensor é projetado para atender a requisitos específicos de medição e operação, e, portanto, é classificado de acordo com várias características-chave. Além da classificação em analógicos, digitais e MEMS, há uma variedade de classificações refletindo outros fatores importantes.

Do ponto de vista das **grandezas físicas mensuráveis**, existe uma ampla variedade de sensores. A capacidade de transduzir variações dessas grandezas em sinais elétricos facilita a integração dos sensores a sistemas embarcados, possibilitando a implementação de soluções automatizadas e a coleta de dados em tempo real. Essa versatilidade e precisão tornam os sensores componentes fundamentais em diversas aplicações tecnológicas e de automação. Entre esses sensores, destacam-se:

- **Sensores de Entrada Pontual**: Este dispositivo é projetado para detectar a localização de um objeto, como a pressão de um botão ou a ativação de uma tecla em um teclado, convertendo essa informação em um sinal elétrico. A transdução ocorre por meio de diferentes princípios físicos, como a variação de resistência (em botões de pressão) ou a mudança na capacitância (em teclados de membrana). Nos **botões de pressão**, a ação de pressionar o botão provoca uma alteração na resistência elétrica, que é medida e convertida em um sinal digital, indicando que a posição “ativada” foi alcançada. Já os teclados de membrana, apresentados no Roteiro 8, funcionam com base na detecção da capacitância entre camadas condutoras; ao pressionar uma tecla, as camadas se conectam, gerando uma alteração na capacitância que é interpretada como um comando. Esses sensores são amplamente utilizados em sistemas embarcados em diversas aplicações, incluindo interfaces de usuário em dispositivos eletrônicos, painéis de controle e equipamentos de automação. Por exemplo, em painéis de controle de eletrodomésticos, botões de pressão permitem ao usuário selecionar funções, enquanto teclados de membrana são comuns em dispositivos portáteis, como calculadoras e controles remotos, proporcionando uma interação fácil e eficiente com o sistema.
- **Sensores de Temperatura**: Esse dispositivo é responsável por medir a temperatura de um ambiente ou objeto e converte essa informação em um sinal elétrico, permitindo a monitorização e o controle de processos térmicos. A transdução ocorre quando o sensor detecta mudanças de temperatura e utiliza efeitos físicos, como a variação da resistência elétrica (em termistores), o efeito termoelétrico (em termopares) ou a alteração na tensão gerada (em sensores de temperatura baseados em diodos). No caso dos **termistores**, a resistência do material varia com a temperatura, permitindo que

pequenas mudanças de temperatura sejam convertidas em mudanças detectáveis na resistência elétrica. Já os **termopare**s operam com base no efeito Seebeck, onde dois metais diferentes, quando unidos, geram uma tensão elétrica proporcional à diferença de temperatura entre as junções. Os **sensores de temperatura baseados em diodos** aproveitam a relação entre a corrente elétrica e a temperatura para fornecer medições precisas. Esses sensores são amplamente utilizados em sistemas embarcados, como em dispositivos de controle de climatização, sistemas de automação residencial e aplicações industriais. Por exemplo, em aquecedores e sistemas de ar condicionado, sensores de temperatura monitoram continuamente as condições ambientais e ajustam o funcionamento do sistema para manter a temperatura desejada. Em ambientes médicos, sensores de temperatura são usados para o monitoramento de pacientes e o controle de equipamentos, garantindo a segurança e eficácia dos tratamentos.

- **Sensores de Pressão**: Este dispositivo é projetado para medir a pressão de gases ou líquidos e converte essa informação em um sinal elétrico, permitindo a monitorização e o controle de processos industriais e ambientais. A transdução ocorre quando o sensor detecta variações na pressão e utiliza efeitos físicos, como a variação de resistência elétrica (em sensores piezoresistivos), a deformação de um diafragma (em sensores piezoelétricos) ou a variação de capacitância (em sensores capacitivos). Nos **sensores piezoresistivos**, a pressão aplicada causa uma mudança na resistência elétrica de um material semicondutor, que pode ser medida e convertida em um sinal elétrico. Os **sensores piezoelétricos** aproveitam o efeito piezoelétrico, onde a pressão aplicada a um material piezoelétrico gera uma tensão elétrica proporcional à força aplicada. Já os **sensores capacitivos** operam com base na variação da capacitância entre placas metálicas, cuja distância muda em resposta à pressão aplicada. Esses sensores são amplamente utilizados em sistemas embarcados em diversas aplicações, como monitoramento de pressão em sistemas hidráulicos, controle de pressão em tanques de combustível, e medições em processos industriais. Por exemplo, em aplicações automotivas, sensores de pressão monitoram a pressão dos pneus para garantir a segurança e eficiência do veículo. Em sistemas de HVAC (do inglês *Heating, Ventilation, and Air Conditioning*), eles ajudam a regular a pressão do ar, otimizando o desempenho do sistema.
- **Sensores de Deslocamento**: Este dispositivo é projetado para medir a posição ou a movimentação de um objeto e converte essa informação em um sinal elétrico, permitindo a monitorização e o controle de sistemas mecânicos e automatizados. A transdução ocorre quando o sensor detecta alterações na posição e utiliza efeitos físicos, como a variação de resistência elétrica (em potenciômetros), a variação de capacitância (em sensores capacitivos), o efeito piezoelétrico ou a variação magnética ou óptica. Nos **potenciômetros**, o deslocamento do eixo gera uma mudança na resistência, que pode ser medida e convertida em um sinal elétrico proporcional à posição. Os **sensores capacitivos** operam com base na variação da capacitância entre placas metálicas, cuja distância muda conforme o deslocamento do objeto. Já os **sensores piezoelétricos** aproveitam o efeito piezoelétrico, onde a deformação mecânica causada pelo deslocamento gera uma tensão elétrica. Os ***encoders* lineares**, por sua vez, são utilizados para medir deslocamentos em linha reta, convertendo essa

informação em sinais elétricos, conforme explicado no Roteiro 6. Esses sensores são amplamente utilizados em sistemas embarcados em diversas aplicações, incluindo dispositivos de entrada, como *joysticks*, conforme apresentado no Roteiro 9. Eles são amplamente aplicados no controle de movimento em robótica, monitoramento de posição em atuadores e medição de deslocamentos em máquinas industriais. Por exemplo, em aplicações automotivas, sensores de deslocamento podem ser utilizados para monitorar a posição de componentes como aceleradores e freios, garantindo uma resposta precisa e segura.

- **Sensores de Vazão**: Este dispositivo é projetado para medir a quantidade de fluido que passa por um ponto específico em um determinado intervalo de tempo, convertendo essa informação em um sinal elétrico. A transdução ocorre quando o sensor detecta a velocidade ou o volume do fluido e utiliza princípios físicos, como a variação de pressão, a capacitância ou a resistência. Os **sensores de vazão por pressão diferencial** medem a diferença de pressão entre dois pontos de um tubo, permitindo calcular a vazão com base na lei de Bernoulli. Os **sensores capacitivos** utilizam a variação da capacitância causada pelo movimento do fluido para determinar a vazão. Já os **sensores de vazão eletromagnéticos** aproveitam o princípio de indução eletromagnética, onde o fluido condutor gera uma tensão proporcional à sua velocidade quando atravessa um campo magnético. Esses sensores são amplamente utilizados em sistemas embarcados em diversas aplicações, como monitoramento de sistemas hidráulicos e pneumáticos, controle de processos industriais e medição de consumo em redes de abastecimento de água. Por exemplo, em aplicações agrícolas, sensores de vazão são essenciais para otimizar o uso de irrigação, garantindo a eficiência do consumo de água.
- **Sensores de Movimento**: Este dispositivo é projetado para detectar a presença ou o deslocamento de objetos em uma determinada área e converte essa informação em um sinal elétrico. A transdução ocorre quando o sensor capta variações no ambiente, utilizando princípios físicos como infravermelho, ultrassom ou variação de capacitância. Os **sensores de movimento infravermelhos** detectam mudanças na radiação infravermelha emitida por objetos quentes, como seres humanos, ativando um sinal elétrico quando detectam movimento. Os **sensores ultrassônicos** emitem ondas sonoras de alta frequência e medem o tempo que leva para essas ondas retornarem após refletirem em um objeto, permitindo calcular a distância e detectar movimento. Já os **sensores capacitivos** detectam mudanças na capacitância provocadas pelo movimento de objetos próximos. Esses sensores são amplamente utilizados em sistemas embarcados em diversas aplicações, como sistemas de segurança, automação residencial e controle de iluminação. Por exemplo, em aplicações de segurança, sensores de movimento podem ativar alarmes ou câmeras de vigilância quando detectam movimento em áreas restritas.
- **Sensores de Nível**: Este dispositivo é projetado para medir a altura ou a quantidade de um líquido ou sólido em um recipiente, convertendo essa informação em um sinal elétrico. A transdução ocorre por meio de diferentes princípios físicos, como flutuação, capacitância, pressão ou ultrassom. Os **sensores de nível por flutuador** utilizam um dispositivo flutuante que se desloca na superfície do líquido; quando a

altura do líquido varia, o flutuador move-se, ativando um mecanismo que gera um sinal elétrico. Os **sensores capacitivos** medem a variação da capacitância entre duas placas, que muda conforme o nível do líquido altera a distância entre elas. Os **sensores de pressão** medem a pressão exercida pela coluna de líquido, convertendo essa pressão em um sinal correspondente ao nível do líquido. Já os **sensores ultrassônicos** emitem ondas sonoras e medem o tempo que leva para elas refletirem na superfície do líquido, permitindo calcular o nível com precisão. Esses sensores são amplamente utilizados em sistemas embarcados em diversas aplicações, como monitoramento de reservatórios, controle de processos industriais e automação de sistemas de irrigação. Por exemplo, em aplicações de abastecimento de água, sensores de nível podem ser utilizados para garantir que tanques não transbordem ou fiquem vazios, ativando bombas conforme necessário.

- **Sensores de Luminosidade**: Este dispositivo é projetado para medir a intensidade da luz em um ambiente, convertendo essa informação em um sinal elétrico. A transdução ocorre através de princípios fotométricos, onde a luz incidente em um material sensível gera uma corrente elétrica ou uma mudança de resistência. Os **sensores fotográficos**, como os fotodiodos e fototransistores, utilizam materiais semicondutores que produzem uma corrente elétrica proporcional à intensidade da luz que os atinge. Quando a luz incide sobre esses sensores, a energia da luz excita os elétrons, resultando em uma corrente elétrica que pode ser medida. Já os **sensores de resistência dependente de luz** (em inglês, *Light Dependent Resistor* – LDR) variam sua resistência elétrica com base na quantidade de luz que recebem; em ambientes mais iluminados, sua resistência diminui, permitindo a passagem de mais corrente. Esses sensores são amplamente utilizados em sistemas embarcados em diversas aplicações, como controle de iluminação automática, monitoramento ambiental e dispositivos de segurança. Por exemplo, em sistemas de iluminação inteligente, sensores de luminosidade podem ajustar a intensidade das luzes com base na quantidade de luz natural disponível, economizando energia e melhorando a eficiência.
- **Sensores de Rotação**: Este dispositivo é projetado para medir a velocidade ou a posição angular de um objeto em movimento, convertendo essa informação em um sinal elétrico. A transdução ocorre por meio de princípios como o efeito Hall, a variação de resistência ou a tecnologia óptica. Os **sensores de efeito Hall** utilizam um campo magnético para detectar a rotação. Quando um ímã é posicionado próximo ao sensor, a variação no campo magnético provoca uma alteração na tensão de saída, que pode ser interpretada como uma medida da rotação. Já os ***encoders* rotativos**, apresentado no Roteiro 6, que podem ser ópticos ou magnéticos, funcionam registrando a passagem de marcas em um disco rotativo, gerando pulsos elétricos que representam a posição angular e a velocidade de rotação. Esses sensores são amplamente utilizados em sistemas embarcados em diversas aplicações, como controle de motores, robótica e automação industrial. Por exemplo, em sistemas de controle de movimento, sensores de rotação podem fornecer *feedback* preciso sobre a posição de um eixo, permitindo ajustes em tempo real para garantir a precisão do movimento.

- **Sensores de Aceleração**: Este dispositivo é projetado para medir a aceleração de um objeto em uma ou mais direções, convertendo essa informação em um sinal elétrico. A transdução ocorre quando o sensor detecta variações na força aplicada, utilizando efeitos físicos, como a variação de capacitância (em acelerômetros capacitivos) ou a variação de resistência (em acelerômetros piezoelétricos). Nos **acelerômetros capacitivos**, como MPU-6065, a aceleração é medida por meio da mudança na capacitância entre placas que se deslocam conforme a força inercial, incluindo a aceleração gravitacional. Já os **acelerômetros piezoelétricos** aproveitam o efeito piezoelétrico, onde a deformação causada pela aceleração gera uma tensão elétrica proporcional à força aplicada. Os principais acelerômetros comerciais utilizam a aceleração da gravidade como referência, o que permite a medição da orientação do sensor em relação ao vetor gravitacional. Isso é particularmente útil em aplicações onde a detecção da inclinação ou a orientação em relação ao solo é crítica, como em dispositivos móveis e sistemas de navegação. Essa referência também melhora a precisão das medições de movimento, uma vez que a gravidade fornece um ponto de partida consistente para as leituras. Esses sensores são amplamente utilizados em sistemas embarcados em diversas aplicações, incluindo dispositivos móveis para detecção de movimento, sistemas de navegação em veículos e monitoramento de condições de funcionamento em equipamentos industriais. Por exemplo, em *smartphones*, os sensores de aceleração são usados para detectar a orientação do dispositivo e ajustar a tela automaticamente. Em aplicações automotivas, podem ser utilizados para monitorar a dinâmica do veículo, contribuindo para a segurança e o desempenho.
- **Sensores de Orientação Espacial**: Este dispositivo é projetado para determinar a orientação de um objeto em relação a um sistema de coordenadas tridimensionais, convertendo essa informação em um sinal elétrico. A transdução ocorre por meio da combinação de diferentes tecnologias, como giroscópios e acelerômetros. Os **giroscópios** medem a taxa de rotação em torno de um ou mais eixos, enquanto os **acelerômetros** detectam a aceleração e, frequentemente, a gravidade. A combinação permite que o sistema compense a gravidade, resultando em uma orientação mais exata, mesmo durante movimentos dinâmicos. Isso é fundamental em aplicações que exigem alta precisão, como em dispositivos móveis, sistemas de navegação e controle de estabilidade em veículos. Em *smartphones*, por exemplo, sensores de orientação espacial ajudam a ajustar a tela automaticamente e a detectar a posição em jogos de realidade aumentada. Em drones e robôs, eles são essenciais para a navegação e controle de movimento, garantindo que o dispositivo mantenha sua trajetória desejada e reaja adequadamente a mudanças de posição e orientação.
- **Sensores de Campo Magnético:** Este dispositivo é projetado para medir a intensidade e a direção de um campo magnético, convertendo essa informação em um sinal elétrico. A transdução ocorre por meio de variações no campo magnético, utilizando efeitos físicos como o efeito Hall, que gera uma tensão elétrica proporcional à força do campo magnético aplicado. Esses sensores são cruciais em diversas aplicações, como navegação e detecção de objetos metálicos. Diferentemente dos sensores de aceleração e orientação espacial, que se concentram em movimentos e

inclinações, os sensores de campo magnético fornecem informações sobre a presença e a intensidade de campos magnéticos ao redor. Isso é particularmente útil em *smartphones*, onde sensores magnéticos auxiliam na orientação do dispositivo, provendo uma referência fixa de direção (norte magnético) e na compensação de movimentos, permitindo uma navegação mais precisa em aplicativos de mapas. Em aplicações automotivas, os sensores de campo magnético são utilizados em sistemas de monitoramento de posição, como sensores de posição de roda e em sistemas de controle de estabilidade. Além disso, esses sensores desempenham um papel vital em ambientes industriais, onde são utilizados para detectar a presença de metais, monitorar sistemas de segurança e em aplicações de IoT (*Internet* das Coisas) para rastreamento e automação.

- **Sensores de Distância**: Este dispositivo é projetado para medir a distância entre o sensor e um objeto, convertendo essa informação em um sinal elétrico. A transdução ocorre por meio de diferentes princípios de operação, como ultrassom, infravermelho ou laser. Nos **sensores ultrassônicos**, uma onda sonora é emitida e, ao ser refletida pelo objeto, o tempo que leva para retornar é medido, permitindo calcular a distância com base na velocidade do som. Já os **sensores de infravermelho** utilizam luz infravermelha para medir a distância, geralmente através da triangulação ou do tempo de voo. **Sensores a laser**, por sua vez, emitem um feixe de luz que é refletido pelo objeto e mede o tempo que leva para retornar, proporcionando alta precisão e capacidade de medição em longas distâncias. Esses sensores são amplamente utilizados em sistemas embarcados em diversas aplicações, incluindo robótica, automação industrial e sistemas de navegação. Por exemplo, em robôs, sensores de distância ajudam na detecção de obstáculos e na navegação autônoma, permitindo que o robô mapeie seu ambiente e evite colisões. Em aplicações de automação, como em linhas de produção, esses sensores podem ser utilizados para medir a posição de objetos e otimizar processos. Além disso, em sistemas de segurança, sensores de distância podem ser empregados para monitorar áreas e detectar intrusões. A versatilidade e a precisão na medição tornam os sensores de distância componentes valiosos em diversas tecnologias modernas.
- **Sensores de Corrente**: Este dispositivo é projetado para medir a intensidade da corrente elétrica que flui em um circuito, convertendo essa informação em um sinal elétrico. A transdução ocorre por meio de diferentes princípios de operação, como o efeito Hall, transformadores de corrente ou resistência shunt. No **caso do efeito Hall**, um campo magnético gerado pela corrente elétrica cria uma tensão que é proporcional à intensidade da corrente, permitindo a medição sem interrupção do circuito. **Transformadores de corrente** operam com base na indução eletromagnética, onde a corrente primária gera um campo magnético que induz uma corrente na bobina secundária, proporcional à corrente original. Já os **sensores de resistência *shunt*** utilizam uma resistência de precisão inserida no circuito, onde a tensão gerada pela passagem da corrente é medida e convertida em um valor de corrente. Esses sensores são amplamente utilizados em sistemas embarcados em diversas aplicações, incluindo monitoramento de consumo de energia, controle de motores e proteção de circuitos. Por exemplo, em sistemas de automação residencial, sensores de corrente ajudam a

monitorar o consumo de energia de eletrodomésticos, permitindo a otimização do uso e a redução de custos. Em aplicações industriais, esses sensores são essenciais para o controle de motores, garantindo que funcionem dentro de limites seguros e eficientes. Além disso, em sistemas de proteção, sensores de corrente podem detectar sobrecargas e curtos-circuitos, ativando mecanismos de segurança para evitar danos aos equipamentos. A precisão e a confiabilidade na medição da corrente elétrica fazem dos sensores de corrente componentes cruciais em diversas tecnologias modernas.

A transdução de uma mesma grandeza física em um sinal elétrico pode ser realizada por meio de diferentes efeitos físicos presentes na natureza. Isso implica que, conforme o fenômeno físico explorado, é possível classificar os sensores de maneira variada. Essa classificação, fundamentada no **princípio de operação**, proporciona aos engenheiros e projetistas a capacidade de selecionar a tecnologia mais adequada às suas necessidades específicas. Cada fenômeno físico possui características únicas que impactam aspectos como precisão, sensibilidade, faixa de operação e resposta a diferentes condições ambientais. Entre os principais efeitos físicos que se destacam, podemos mencionar:

- **Efeito Fotovoltaico:** O efeito fotovoltaico é o fenômeno pelo qual materiais semicondutores geram uma corrente elétrica quando expostos à luz. Ao absorver a radiação luminosa, os fótons excitam elétrons em um semicondutor, criando pares de elétron-lacuna que podem ser mobilizados para gerar eletricidade. Sensores baseados no efeito fotovoltaico são particularmente apropriados para aplicações em que a detecção de luz é essencial. Um exemplo são os sensores de luz, que são utilizados em sistemas de automação, como iluminação inteligente, onde a intensidade da luz é monitorada para ajustar o brilho das lâmpadas. Além disso, as células solares, embora não sejam sensores no sentido tradicional, empregam o efeito fotovoltaico para capturar luz e convertê-la em eletricidade, sendo amplamente utilizadas em sistemas de energia renovável. Outro uso importante é em sensores de radiação, que são aplicados em estações meteorológicas e equipamentos de pesquisa para medir a intensidade da radiação solar e monitorar condições ambientais. Por fim, os sensores de imagem em câmeras digitais também utilizam o efeito fotovoltaico para converter luz em sinais elétricos, permitindo a captura de fotografias.
- **Efeito Fotoelétrico (Efeito Hertz)**: O efeito fotoelétrico é o fenômeno pelo qual elétrons são ejetados de um material quando este é iluminado por radiação eletromagnética, geralmente luz visível ou ultravioleta. Quando a energia dos fótons da luz incide sobre a superfície de um material, como um metal, ela pode ser suficiente para superar a função trabalho do material, liberando elétrons e gerando uma corrente elétrica. Sensores baseados no efeito fotoelétrico são especialmente adequados para várias aplicações. Um exemplo são as células fotográficas, que são usadas em dispositivos como fotômetros e sensores de luminosidade, detectando a intensidade da luz e ajustando sistemas de iluminação, como em painéis solares. Outro uso importante é em sensores de presença, empregados em sistemas de segurança e automação, onde a detecção de movimento ocorre através da interrupção de um feixe de luz, acionando alarmes ou sistemas de iluminação. Além disso, leitores de códigos

de barras utilizam o efeito fotoelétrico para detectar a reflexão da luz em diferentes intensidades ao passar sobre as barras do código, permitindo a leitura e interpretação de dados.

- **Efeito Piezoelétrico**: O efeito piezoelétrico é um fenômeno em que certos materiais, como cristais e cerâmicas piezoelétricas, geram uma tensão elétrica quando submetidos a pressão mecânica. Essa propriedade é resultado do rearranjo das cargas elétricas em sua estrutura cristalina, que ocorre sob estresse mecânico. Sensores baseados no efeito piezoelétrico são adequados para várias aplicações, incluindo sensores de pressão, que são utilizados em sistemas de monitoramento de pressão em diversas indústrias, permitindo medições precisas em ambientes variáveis. Os microfones piezoelétricos também desempenham um papel importante, pois convertem ondas sonoras em sinais elétricos e são amplamente usados em dispositivos de áudio. Além disso, sensores piezoelétricos são empregados em impressoras 3D para controlar a movimentação e a deposição de material, garantindo precisão na impressão. Esses sensores também são aplicados em dispositivos de alerta e sinalização, onde a detecção de pressão ou vibração ativa um sinal de alerta, e em sistemas de controle de movimento, especialmente na robótica e automação, onde os atuadores piezoelétricos proporcionam movimentos precisos, essenciais em aplicações que exigem controle fino.
- **Efeito Resistivo**: O efeito resistivo é baseado na variação da resistência elétrica de um material em resposta a mudanças em fatores como pressão, temperatura ou deslocamento. Quando um material condutor ou semicondutor é submetido a uma força ou condição externa, sua resistência elétrica pode mudar, permitindo a conversão dessa variação em um sinal elétrico. Sensores que utilizam o efeito resistivo são amplamente empregados em diversas aplicações. Por exemplo, potenciômetros são utilizados em controles de volume e ajuste de intensidade, onde a posição do eixo altera a resistência e, consequentemente, o sinal elétrico. Sensores de pressão resistivos são usados em sistemas industriais para monitorar a pressão de fluidos, enquanto sensores de entrada pontual, como os utilizados em botões e teclados, detectam a ativação de comandos por meio da variação da resistência quando pressionados. Esse efeito é particularmente útil em aplicações onde medições precisas de força, posição ou pressão são necessárias, tornando os sensores resistivos componentes essenciais em sistemas de automação, controle e interface homem-máquina.
- **Efeito de Indução Mútua**: O efeito de indução mútua ocorre quando a corrente elétrica que flui através de um condutor cria um campo magnético, que, por sua vez, induz uma corrente em um segundo condutor próximo. Esse fenômeno é fundamental na operação de transformadores e em dispositivos de medição de corrente, onde a variação da corrente em um condutor influencia a corrente em outro, permitindo a detecção de mudanças elétricas. Sensores que utilizam o efeito de indução mútua são comumente empregados em diversas aplicações. Por exemplo, em transformadores, esse efeito é utilizado para transmitir energia elétrica entre circuitos, ajustando a tensão conforme necessário. Além disso, sensores de corrente que operam com indução mútua são utilizados em sistemas de monitoramento de energia, permitindo

medir a corrente em fios sem contato direto, o que proporciona segurança e precisão em aplicações industriais e residenciais. Também é possível encontrar esse efeito em transdutores de posição, onde a variação da posição de um objeto afeta o campo magnético, gerando sinais elétricos que podem ser medidos. Esses sensores são especialmente úteis em ambientes onde a medição não intrusiva é desejável, como em sistemas de automação industrial, medição de energia elétrica e monitoramento de equipamentos, garantindo eficiência e segurança.

- **Efeito Eletromagnético:** O efeito se refere à interação entre campos elétricos e magnéticos, resultando na geração de força eletromotriz (FEM) em condutores expostos a variações de campos magnéticos. Esse fenômeno é fundamental em dispositivos como geradores e motores elétricos, onde a energia elétrica é convertida em energia mecânica e vice-versa. Sensores que utilizam o efeito eletromagnético são amplamente aplicados em diversas áreas. Por exemplo, em sensores de proximidade, eles detectam a presença de objetos metálicos sem contato, sendo muito usados em sistemas de automação e controle industrial. Além disso, sensores de corrente baseados nesse efeito podem monitorar a intensidade da corrente elétrica em circuitos, sendo essenciais para sistemas de medição de energia. A principal diferença entre o efeito eletromagnético e o efeito de indução magnética é que, enquanto o efeito de indução magnética se concentra na indução de corrente em um condutor devido à variação de um campo magnético, o efeito eletromagnético engloba uma interação mais ampla entre campos elétricos e magnéticos que resulta em movimento ou geração de energia. Essa distinção é crucial para a aplicação adequada em diferentes tecnologias e dispositivos.
- **Efeito Hall**: O efeito é um fenômeno físico que ocorre quando um condutor ou semicondutor é exposto a um campo magnético perpendicular à direção da corrente elétrica que flui através dele. Esse efeito resulta na geração de uma tensão elétrica (tensão Hall) nas extremidades do material, perpendicular tanto à corrente quanto ao campo magnético aplicado. Essa tensão pode ser medida e utilizada para determinar a intensidade do campo magnético. Sensores baseados no efeito Hall são amplamente utilizados em várias aplicações, como em sistemas de detecção de corrente elétrica, onde permitem medir a intensidade da corrente sem necessidade de contato direto com o circuito. Também são utilizados em sensores de posição e velocidade, como em motores e acionamentos, além de serem comuns em dispositivos automotivos para detecção de posição de componentes como eixos e rotores. A diferença entre o efeito Hall e os efeitos de indução magnética e eletromagnético está em sua abordagem e aplicação. O efeito Hall é específico para a geração de uma tensão elétrica em resposta a um campo magnético aplicado a um condutor, enquanto o efeito de indução magnética se refere à geração de corrente em um circuito fechado devido à variação de um campo magnético. Já o efeito eletromagnético abrange uma interação mais ampla entre campos elétricos e magnéticos, sendo fundamental para a operação de motores e geradores. Assim, o efeito Hall é particularmente útil para medições e detecções precisas em ambientes onde a interação magnética é necessária.
- **Efeito Capacitivo**: O efeito se refere à capacidade de um capacitor de armazenar carga elétrica, que varia com a distância entre as placas do capacitor ou com a área das

placas. Em sensores capacitivos, essa variação é utilizada para detectar mudanças na presença ou na posição de objetos, bem como em medições de pressão e umidade. Quando um objeto se aproxima do sensor, a capacitância muda, permitindo que essa alteração seja convertida em um sinal elétrico. Sensores capacitivos são amplamente utilizados em diversas aplicações, como em telas sensíveis ao toque, detectores de proximidade, e medidores de nível de líquidos. Sua capacidade de funcionar sem contato físico os torna ideais para ambientes onde a higiene é importante, como em equipamentos médicos e em indústrias alimentícias. Além disso, são frequentemente empregados em sistemas automotivos e de automação industrial para detectar a presença ou movimento de objetos.

- **Efeitos Químicos**: O efeito se refere à interação entre substâncias químicas que resulta em mudanças físicas ou elétricas, frequentemente utilizadas em sensores para detectar a presença e a concentração de determinados compostos. Esses sensores funcionam com base em reações químicas que geram um sinal elétrico proporcional à quantidade de substância analisada. Sensores baseados no efeito químico são amplamente utilizados em aplicações como monitoramento da qualidade do ar, onde detectam poluentes e gases tóxicos, como dióxido de carbono e monóxido de carbono. Eles também são essenciais em análises químicas e bioquímicas, sendo utilizados em dispositivos de diagnóstico médico, como sensores de glicose, que monitoram os níveis de açúcar no sangue. Além disso, encontram aplicação em processos industriais para controle de reações químicas e em laboratórios para medições precisas de concentrações de substâncias. A capacidade de fornecer medições em tempo real torna esses sensores valiosos em diversas áreas, incluindo saúde, segurança e meio ambiente.

Igualmente importante para projeto de circuitos de interface entre os sensores e os microcontroladores é a demanda por **uma fonte de alimentação** externa para identificar as variações das grandezas físicas de interesse. Quanto a isso, os sensores podem ser classificados em:

- **Ativos** ou geradores: Geram um sinal elétrico de saída em resposta a um estímulo sem uma fonte de alimentação externa. Células fotovoltáicas, cristais piezoelétricos e termopares são exemplos de sensores ativos.
- **Passivos** ou moduladores: Precisam ser excitados por uma fonte de alimentação externa para gerar um sinal elétrico de saída. Potenciômetros são exemplos de sensores resistivos passivos.

Classificação de atuadores

Atuadores são componentes essenciais em sistemas embarcados e automação, pois convertem sinais elétricos de controle em ações físicas, como movimento, força ou alteração de estado. Compreender a taxonomia dos atuadores permite que os desenvolvedores identifiquem rapidamente as opções disponíveis, considerando fatores como o tipo de transdução, a tecnologia empregada (elétrica, hidráulica, pneumática, etc.) e a fonte de alimentação. Essa organização simplifica a seleção do atuador mais adequado para cada aplicação

Do ponto de visto do **tipo de transdução a partir de energia elétrica**, existe uma diversidade de transduções que os materiais naturais podem realizar, permitindo uma ampla gama de aplicações tecnológicas. Aqui estão algumas delas mais conhecidas:

- **Efeito de Campo Elétrico**: Este fenômeno ocorre quando a aplicação de um campo elétrico em certos cristais e materiais cerâmico, denominados piezoelétricos, resulta em deformação mecânica. A interação entre o campo elétrico e a estrutura interna do material provoca uma mudança em sua forma ou volume. Esse efeito é amplamente utilizado em atuadores e sensores, onde a precisão na movimentação ou na detecção de variações é crucial. Por meio do controle do campo elétrico aplicado, é possível ajustar com precisão a deformação do material, tornando-o adequado para aplicações em dispositivos que requerem movimentação controlada, como em sistemas de microeletrônica e tecnologia médica. Além disso, os materiais piezoelétricos têm a capacidade de gerar uma tensão elétrica quando submetidos a pressão mecânica, fenômeno conhecido como **efeito piezoelétrico,** que é utilizado em dispositivos como buzzers.
- **Efeito Eletroluminescente**: Este fenômeno consiste na emissão de radiações visíveis ou invisíveis em resposta a uma corrente elétrica ou a um forte campo elétrico. Ele é o inverso do **efeito fotoelétrico**, onde a luz incidente em um material gera uma corrente elétrica, sendo esse o princípio fundamental das células solares e fotodetectores. Os materiais que exibem o efeito eletroluminescente, como semicondutores, convertem a energia elétrica em luz. Um exemplo prático desse efeito é o LED (do inglês *Light Emitting Diode*), que transforma a energia elétrica em luz visível de maneira eficiente e é amplamente utilizado em aplicações que vão desde iluminação até *displays* eletrônicos. O controle preciso da corrente elétrica permite a modulação da intensidade da luz, tornando os LEDs versáteis para uma variedade de usos em tecnologia e comunicação.
- **Efeito Joule (ou Efeito Termoelétrico)**: Este fenômeno estabelece a relação entre a produção de calor e a corrente elétrica que percorre um condutor ao longo do tempo. Quando uma corrente elétrica flui através de um resistor, parte da energia elétrica é convertida em calor devido à resistência do material. Um exemplo comum é o uso de resistores feitos de ligas metálicas de alta resistividade, como a liga de nicromo, que contém de 15% a 25% de cromo e de 19% a 80% de níquel, com o restante composto por ferro. Esses materiais conseguem transformar quase integralmente a energia elétrica em calor, tornando-os ideais para aplicações de aquecimento.
- **Efeito de Arco Elétrico**: Este fenômeno ocorre quando uma corrente elétrica passa através de um gás, gerando arcos elétricos que emitem radiações visíveis, ou luz. Esse efeito é frequentemente observado em lâmpadas fluorescentes, onde a corrente elétrica ioniza o gás dentro do tubo, criando um arco elétrico que produz luz. A emissão luminosa resulta da excitação dos átomos do gás, que liberam energia na forma de radiação visível quando retornam ao seu estado fundamental. Esse efeito é utilizado em diversas aplicações, incluindo iluminação e soldagem, devido à sua capacidade de gerar intensas fontes de luz. Note que a luz é uma parte essencial do processo de soldagem, pois contribui tanto para o aquecimento necessário quanto para a limpeza

da superfície ou a adição de materiais de enchimento, melhorando a qualidade da solda em alguns processos.

- **Efeito Eletromagnético**: Este fenômeno se refere à interação entre campos elétricos e magnéticos, onde a variação de um campo pode induzir o outro. Esse princípio é fundamental em uma variedade de dispositivos, como motores elétricos e geradores, onde a movimentação de cargas elétricas em um campo magnético gera força motriz ou, inversamente, a mudança de um campo elétrico pode criar um campo magnético. O efeito eletromagnético é utilizado em tecnologias como transformadores, onde a energia é transferida entre circuitos elétricos através de campos magnéticos, e em dispositivos de comunicação, como antenas, que convertem sinais elétricos em ondas eletromagnéticas. Esse efeito é essencial para a geração e transmissão de energia elétrica, além de ser a base para muitas inovações tecnológicas.
- **Efeito Magnetoresistivo ou Magnetoelétrico**: Este fenômeno ocorre em determinados materiais que apresentam uma variação na resistência elétrica quando expostos a um campo magnético. Essa mudança na resistência é decorrente da interação entre o campo magnético e os portadores de carga no material. O efeito magnetoresistivo é amplamente utilizado em sensores magnéticos e em dispositivos de armazenamento, como discos rígidos, onde a leitura dos dados é realizada através da detecção das alterações na resistência elétrica. Graças à sua capacidade de resposta a campos magnéticos, esse efeito permite a miniaturização e o aumento da capacidade de armazenamento em tecnologias modernas. Assim, materiais que exibem esse efeito são especialmente valiosos em aplicações que exigem precisão e sensibilidade, como em sistemas de navegação e monitoramento industrial.

Especificamente, os **atuadores que geram movimentos físicos** podem ser classificados em diferentes categorias com base em seus **princípios de operação**. Esses dispositivos funcionam como transdutores, convertendo diversas formas de energia, como elétrica, hidráulica ou pneumática, em energia mecânica (movimento). Essa conversão é essencial em aplicações industriais e tecnológicas, onde o funcionamento de máquinas e sistemas automatizados é baseado em transformação eficiente de uma forma de energia em outra. É importante destacar que, embora os atuadores possam operar com várias fontes de energia, o controle dessas formas de energia é frequentemente realizado por meio de sistemas digitais, incluindo microcontroladores. Essa abordagem permite uma integração mais eficaz e precisa em sistemas eletrônicos, facilitando o desenvolvimento de soluções automatizadas que atendem a requisitos específicos de desempenho e eficiência. Assim, compreender os diferentes tipos de atuadores de movimento é fundamental para otimizar o funcionamento e a eficácia de sistemas automatizados. Cada classe de atuador possui características, vantagens e desvantagens específicas, que influenciam a escolha de acordo com as necessidades da aplicação. As principais classes incluem:

- **Atuadores Elétricos**: Esses dispositivos, que incluem motores e solenóides, convertem energia elétrica diretamente em movimento mecânico e são amplamente utilizados em sistemas microcontrolados devido à sua facilidade de controle. Os **motores elétricos** são projetados para converter energia elétrica em movimento

rotativo. Eles são onipresentes em uma variedade de aplicações, como eletrodomésticos, veículos elétricos e máquinas-ferramenta. Sua capacidade de oferecer controle preciso de velocidade e torque os torna ideais para tarefas que exigem movimentação contínua. Os **solenóides**, por outro lado, transformam energia elétrica em movimento linear. Esse tipo de atuador é frequentemente utilizado em sistemas de travamento, válvulas e como componentes de controle em equipamentos automáticos. Sua operação simples e rápida os torna eficientes para aplicações que requerem movimentos curtos e precisos. Ambos os tipos de atuadores elétricos são valorizados pela sua versatilidade e facilidade de integração em sistemas eletrônicos, permitindo um controle eficaz em diversas aplicações industriais e comerciais.

- **Atuadores Pneumáticos**: Esses dispositivos operam com ar comprimido, oferecendo movimentos rápidos e uma excelente relação entre força e peso, sendo amplamente utilizados em diversas aplicações industriais. Os **cilindros pneumáticos** são um exemplo clássico, utilizando ar comprimido para criar movimento linear. Esses atuadores são comuns em sistemas de automação e robótica, onde a velocidade e a precisão são essenciais. Sua capacidade de gerar força considerável em relação ao seu peso os torna ideais para tarefas que requerem movimentação rápida e eficiente. Os **motores pneumáticos** geram movimento rotativo utilizando ar comprimido. Eles são utilizados em uma variedade de aplicações industriais, como máquinas de montagem e ferramentas pneumáticas. Esses motores são valorizados por sua durabilidade e alta potência, sendo uma escolha popular em ambientes que exigem operação contínua e resistência. Ambos os tipos de atuadores pneumáticos são fundamentais em sistemas que exigem alta velocidade e resposta rápida, destacando-se pela sua eficiência e capacidade de manuseio em processos automatizados.
- **Atuadores Hidráulicos**: Esses dispositivos utilizam fluidos pressurizados para gerar movimento mecânico, oferecendo alta força e precisão, sendo essenciais em diversas aplicações industriais. Os **cilindros hidráulicos** são um exemplo notável, utilizando fluidos pressurizados, geralmente óleo ou água, para criar movimento linear. Eles são amplamente empregados em máquinas industriais, como prensas hidráulicas, onde a necessidade de força robusta e controle preciso é essencial. Esses cilindros são projetados para suportar cargas pesadas e operar em condições severas. Os **motores hidráulicos** convertem a energia hidráulica em movimento rotativo. Esses motores são comumente utilizados em veículos pesados, como escavadeiras e guindastes, onde a capacidade de gerar força significativa em espaços compactos é essencial. Sua operação eficiente e durabilidade os tornam ideais para aplicações que exigem movimentação constante e confiável. Ambos os tipos de atuadores hidráulicos são fundamentais em sistemas que necessitam de potência e controle precisos, destacando-se pela sua eficácia em aplicações industriais pesadas e em maquinário de grande porte.
- **Atuadores Mecânicos**: Esses dispositivos dependem de mecanismos físicos, como molas ou engrenagens, para gerar movimento. Eles operam com base em princípios mecânicos, aproveitando a força e a resistência de materiais para realizar tarefas específicas. Um exemplo comum de atuador mecânico é o **parafuso de avanço**, que converte movimento rotativo em movimento linear. Esse tipo de atuador é

frequentemente utilizado em máquinas-ferramenta e dispositivos de posicionamento, onde a precisão e o controle são essenciais. Através do giro do parafuso, é possível mover uma carga de forma controlada, permitindo ajustes finos em diversas aplicações. Os atuadores mecânicos são valorizados pela sua simplicidade e robustez, sendo uma escolha ideal para sistemas onde a eletricidade não é desejada ou onde a simplicidade do *design* é prioritária. Sua operação é frequentemente direta e confiável, tornando-os adequados para uma ampla gama de aplicações, desde a automação industrial até dispositivos de uso cotidiano.

- **Atuadores Piezoelétricos**: Esses dispositivos utilizam o efeito piezoelétrico para converter tensão elétrica em deformação mecânica, possibilitando movimentos altamente precisos. Essa capacidade de gerar pequenas mudanças de forma em resposta a variações de tensão os torna ideais para aplicações que exigem controle fino. Um exemplo notável de atuador piezoelétrico é encontrado em **microfones**, onde a deformação mecânica gerada pelas ondas sonoras é convertida em sinal elétrico, permitindo a captura de som com alta fidelidade. Outro uso comum é nas **impressoras 3D**, onde os atuadores piezoelétricos controlam a extrusão do material, possibilitando a criação de objetos com detalhes complexos. Os atuadores piezoelétricos são altamente valorizados em áreas que requerem rapidez e precisão, como em dispositivos médicos, sistemas de controle de movimento e tecnologia de áudio. Sua capacidade de resposta rápida e compacta os torna essenciais em aplicações modernas que demandam tecnologia avançada e desempenho confiável.
- **Atuadores Térmicos**: Esses dispositivos funcionam com base na expansão térmica de materiais quando aquecidos, gerando movimento. Essa propriedade de expansão permite que os atuadores térmicos realizem movimentos lineares ou rotativos em resposta a variações de temperatura. Um exemplo típico de atuador térmico é encontrado em **termostatos**, onde a expansão de um material bimetálico, composto por duas ligas metálicas diferentes, provoca o movimento de um interruptor. Esse mecanismo é amplamente utilizado em sistemas de controle de temperatura, como aquecedores e sistemas de refrigeração, permitindo a manutenção de temperaturas desejadas de forma automática. Os atuadores térmicos são valorizados por sua simplicidade e eficácia, sendo uma escolha confiável para aplicações que requerem resposta a mudanças de temperatura. Sua operação passiva e não eletrônica os torna ideais para ambientes onde a eletricidade pode não estar disponível ou ser desejada, destacando-se em sistemas de automação e controle ambiental.

Com base na quantidade de energia que consomem ou na **potência de entrada necessária** para sua operação, os atuadores podem ser classificados em duas categorias: atuadores de baixa potência e atuadores de alta potência, Essa classificação ajuda na identificação de quais atuadores são mais adequados para diferentes aplicações, especialmente em sistemas microcontrolados, onde o consumo de energia é uma consideração crítica. Os **atuadores de baixa potência** são projetados para operar com sinais elétricos de baixa potência, incluindo atuadores piezoelétricos e atuadores eletromagnéticos de baixa potência. Eles são ideais para aplicações onde a eficiência energética é fundamental, como em dispositivos portáteis, sensores alimentados por baterias e sistemas autônomos.

Em contraste, os **atuadores de alta potência** consomem mais energia e são alimentados por fontes de energia robustas, como motores elétricos, atuadores hidráulicos e pneumáticos. Esses atuadores são mais apropriados para aplicações que exigem movimentos de alta força ou torque, como máquinas industriais, veículos e sistemas de automação. Ao projetar um circuito de acionamento para interface com um microcontrolador, é essencial considerar as características específicas do atuador em questão. Atuadores de alta potência exigem um projeto de circuito cuidadoso devido às suas demandas de energia e controle. Eles frequentemente requerem correntes elétricas mais elevadas e tensões mais altas do que os dispositivos de baixa potência, além de circuitos de controle especializados para garantir um acionamento eficiente e seguro. Adicionalmente, é fundamental implementar medidas de proteção contra sobrecargas e curto-circuitos, bem como técnicas de controle de velocidade, direção e posicionamento, conforme as necessidades da aplicação.

Calibração

A calibração de sensores e atuadores é uma etapa fundamental no desenvolvimento e manutenção de sistemas de medição e controle de alta precisão. Ela envolve a comparação dos valores de saída dos sensores e ações dos atuadores com um padrão de referência de alta exatidão para determinar qualquer desvio ou erro na medição ou no controle. Essa análise fornece informações críticas sobre o desempenho real dos componentes e ajuda a garantir que suas saídas sejam confiáveis. Mesmo componentes de excelente qualidade podem sofrer pequenas variações ao longo do tempo ou devido a condições ambientais, o que pode afetar a **exatidão** ou a **repetibilidade das medições e das ações**. Portanto, é fundamental que os usuários considerem a necessidade de calibração e avaliem os métodos disponíveis para ajustar e otimizar seus sensores e atuadores, mesmo quando a função de calibração não está disponível diretamente no dispositivo.

A facilidade de calibração e a necessidade de ajustes periódicos são considerações importantes na escolha de um sensor ou atuador. A disponibilidade da função de calibração em sensores e atuadores pode variar significativamente com base no fabricante e na faixa de preço do componente. Enquanto sensores e atuadores mais sofisticados e caros tendem a incluir essa função, dispositivos mais acessíveis podem não oferecer a capacidade de calibração integrada. No entanto, **a importância da calibração é universal e se aplica a todos os tipos de sensores e atuadores, independentemente de seu custo**.

Conexão dos sensores e atuadores com microcontroladores

A conexão de um microcontrolador a sensores e atuadores é uma tarefa que envolve diversas estratégias, desde o uso de entradas e saídas analógicas e digitais até a implementação de comunicação serial e sem fio. Compreender essas abordagens pode resultar em projetos mais eficientes e funcionais, permitindo uma interação eficaz com o mundo físico.

A maneira mais simples de conexão é por meio de entradas e saídas analógicas e digitais, como vimos nos roteiros anteriores. As entradas e saídas digitais de propósito geral só podem assumir valores do terra ou a tensão máxima, representando estados binários 0 e 1. Em

contrapartida, as entradas e saídas analógicas variam continuamente entre 0 e uma tensão máxima, dependendo do sensor ou atuador utilizado. Por exemplo, uma saída digital pode controlar um LED, que acende ou apaga com base na tensão aplicada. Já uma entrada analógica pode ser usada para ler o valor de um potenciômetro, que altera sua resistência conforme é girado. Para implementar essas funcionalidades, os Roteiros 1 a 7 mostram que é necessário conectar as entradas e saídas digitais aos pinos digitais de propósito geral do microcontrolador e utilizar as funções do módulo GPIO para manipulá-los no código. Por outro lado, as entradas e saídas analógicas devem ser conectadas aos pinos analógicos, como exploramos no Roteiro 9, e usar as funções dos módulos ADC e DAC para processarmos, respectivamente, as entradas e as saídas no nosso código.

Ligar sensores e atuadores diretamente aos pinos digitais GPIO ou aos pinos ADC e DAC de um microcontrolador pode parecer uma solução simples, mas apresenta várias desvantagens que comprometem a eficiência e a escalabilidade do sistema. Em primeiro lugar, essa abordagem não é flexível, pois limita a capacidade de expandir o sistema. À medida que mais dispositivos são adicionados, a complexidade aumenta, tornando o gerenciamento e a manutenção mais difíceis. Além disso, a comunicação direta pode levar a problemas de interferência e ruído, especialmente em sistemas que operam em ambientes ruidosos, o que pode resultar em leituras imprecisas e falhas na execução de ações. Uma solução de comunicação mais estruturada traz diversas vantagens.

Em primeiro lugar, permite a utilização de protocolos específicos que garantem a integridade e a eficiência da troca de dados. Protocolos como I2C e SPI, por exemplo, possibilitam a comunicação entre múltiplos dispositivos com menos fios e com um controle mais eficaz sobre a troca de informações. Isso não apenas simplifica a fiação, mas também facilita a detecção de erros e a sincronização de dados. Além disso, uma comunicação estruturada permite a implementação de mecanismos de controle mais sofisticados, como a diferenciação entre dispositivos mestres e escravos, que organizam a troca de informações e evitam colisões de dados. Isso é especialmente importante em sistemas complexos onde múltiplos sensores e atuadores devem interagir simultaneamente. Outro ponto a favor de uma abordagem mais organizada é a capacidade de integrar diferentes tipos de sinais e protocolos, o que resulta em maior versatilidade para adaptar o sistema a novas necessidades e tecnologias. Por fim, a utilização de interfaces de comunicação bem definidas melhora a modularidade do sistema, permitindo que componentes sejam facilmente substituídos ou atualizados sem a necessidade de grandes reconfigurações. Isso não apenas economiza tempo e recursos, mas também aumenta a longevidade do sistema.

Portanto, embora a conexão direta possa parecer uma solução rápida, optar por uma comunicação mais estruturada entre um microcontrolador e os sensores/atuadores é fundamental para garantir a eficiência, a escalabilidade e a robustez de sistemas embarcados. Os microcontroladores modernos suportam uma série de protocolos de comunicação serial bem definidos, como I2C, SPI, UART/USART e CAN, que facilitam a implementação dessas comunicações, permitindo a integração de múltiplos dispositivos de forma organizada e eficiente. Esses protocolos não apenas melhoram a confiabilidade da troca de dados, mas

também oferecem flexibilidade para atender às necessidades específicas de cada aplicação, simplificando o design do sistema e reduzindo o tempo de desenvolvimento.

Além disso, é importante considerar a proteção do microcontrolador contra sobretensões e ruídos. O uso de componentes como resistores, capacitores e diodos é fundamental nesse contexto. Implementar resistores de *pull-up* ou *pull-down* garante um estado de entrada estável, enquanto interrupções permitem que o microcontrolador responda a eventos sem a necessidade de polling constante. Também é possível utilizar um resistor e um capacitor para formar um filtro passa-baixa, que suaviza a entrada analógica de sensores ruidosos, melhorando a precisão das medições.

Programação

Muitos sensores e atuadores modernos já vêm com controladores integrados, o que traz várias vantagens. Essa integração simplifica a instalação, eliminando a necessidade de *hardware* adicional, o que facilita a configuração e aumenta a confiabilidades. Com menos conexões externas, o risco de falhas de comunicação diminui, tornando os sistemas mais robustos. Além disso, a comunicação direta entre sensores e atuadores se torna mais eficiente, reduzindo latências e melhorando a precisão. O controlador pode processar dados localmente, permitindo respostas mais rápidas a mudanças nas condições, o que otimiza o desempenho. Essa abordagem também pode resultar em redução de custos, já que menos componentes externos significam menor custo de materiais e ocupação de espaço.

Muitos desses controladores possuem registradores de controle, dados e estado, permitindo que desenvolvedores ajustem e personalizem o funcionamento desses dispositivos por meio da **programação**, de forma análoga à programação dos módulos de microcontroladores. Essa prática de programação permite que os desenvolvedores definam parâmetros específicos, como intervalos de leitura de dados, limites de operação e modos de resposta, adaptando os sensores e atuadores às necessidades particulares de cada aplicação. Por exemplo, ao programar um sensor de temperatura, o desenvolvedor pode especificar a frequência com que as medições são feitas e determinar como os dados devem ser processados e transmitidos.

Além disso, muitos controladores oferecem interfaces de programação (em inglês, *Application Programming Interface* – API) e bibliotecas específicas que facilitam a integração com diversas linguagens e ambientes de desenvolvimento. Essas bibliotecas disponibilizam funções que simplificam a leitura e configuração de sensores e atuadores. Por exemplo, a biblioteca DHT é projetada para ler temperatura e umidade de sensores DHT, enquanto a biblioteca Servo permite o controle preciso da posição de um motor servo. Essa abordagem não só facilita a implementação de algoritmos complexos, capazes de analisar dados em tempo real e tomar decisões com base em condições específicas, mas também possibilita a integração de funcionalidades de aprendizado de máquina. Além disso, a programação de protocolos de comunicação personalizados assegura que os sensores e atuadores se conectem de maneira harmoniosa em redes mais amplas, como a Internet das Coisas (IoT). Com essa flexibilidade, os desenvolvedores podem criar soluções mais

eficientes, escaláveis e adaptáveis, otimizando o desempenho do sistema e aprimorando a interação com os usuários finais.

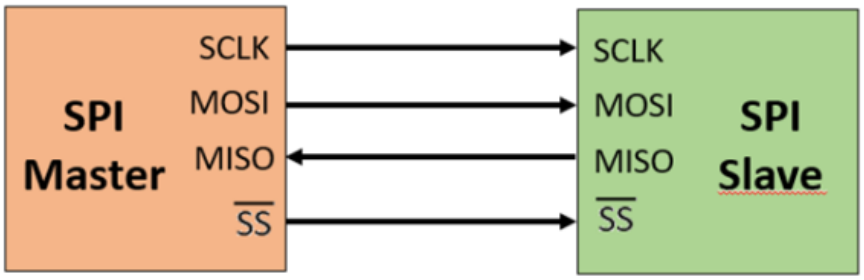
COMUNICAÇÃO SERIAL SÍNCRONA

A comunicação serial síncrona é fundamental para a interação eficiente entre microcontroladores e periféricos em sistemas embarcados. Nesta seção, exploraremos dois dos protocolos mais utilizados: SPI (do inglês *Serial Peripheral Interface*) e I2C (do inglês *Inter-Integrated Circuit*). Ambos oferecem soluções robustas para a troca de dados, mas cada um possui características específicas que os tornam adequados para diferentes aplicações. O SPI se destaca por sua alta velocidade e simplicidade, ideal para sistemas que requerem transferência rápida de dados. Já o I2C é conhecido por sua capacidade de conectar múltiplos dispositivos em um único barramento, facilitando a comunicação em redes mais complexas. Vamos analisar como esses protocolos funcionam, suas vantagens e desvantagens.

Interface serial SPI

A ***Serial Peripheral Interface*** (**SPI**) é um barramento de quatro fios inventado pela Motorola. É composto por uma linha de clock (**SCLK**), uma linha de saída do mestre/entrada do escravo (**MOSI**, do inglês *Master Output-Slave Input*), uma linha de entrada do mestre/saída do escravo (**MISO**, do inglês *Master Input-Slave Output*) e uma linha de seleção de dispositivo (**SS**, do inglês *Slave Select*). Essa última linha pode receber outros nomes, como *Chip Select* (**CS**) ou *Chip Enable* (**CE**).

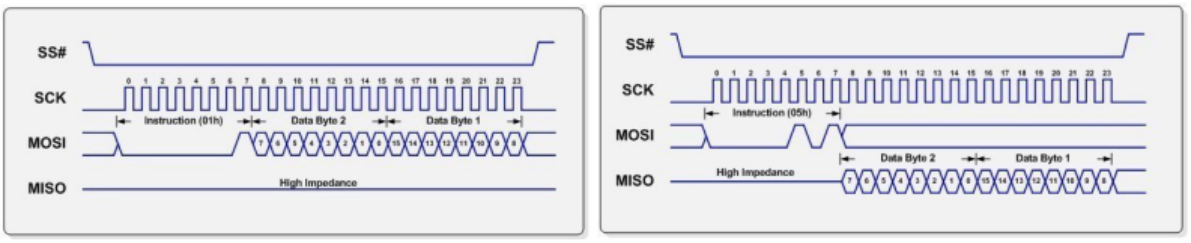
A faixa de velocidade do barramento SPI é muito maior do que a encontrada no I2C ou na Serial Assíncrona; velocidades de até 80 MHz não são incomuns. Existem variantes que fornecem vários *bits* em paralelo para a transferência (até 4 *bits*). Esses *bits* de dados adicionais aumentam o desempenho. O barramento SPI é usado, por exemplo, para conectar a memória *flash* serial que fornece o código de inicialização inicial (***boot***) para plataformas Intel.



O barramento SPI é considerado um barramento ***full-duplex***. Ele utiliza um protocolo síncrono em que os dados são transmitidos e recebidos simultaneamente, em sincronia com o sinal de relógio SCLK gerado pelo mestre, usando dois fios distintos, MISO (*Master In Slave*

Out) e MOSI (*Master Out Slave In*). Quando o mestre transmite um *bit* pela linha MOSI, ele simultaneamente recebe um *bit* correspondente pela linha MISO. Diferentemente de protocolos como o serial assíncrono ou o I2C, o SPI não inclui *bits* de *START*, *STOP* ou *ACK* específicos. O SPI usa um pino de controle dedicado (SS), como um pino Chip Select (CS), para indicar o início e o término de uma transação.

A transferência começa quando o mestre ativa o pino de seleção de escravo (do inglês, *Chip Select* - CS) para o dispositivo com o qual deseja se comunicar. Após a ativação do /CS (geralmente em nível lógico baixo, por isso usaremos a notação com uma "/" na frente do nome do sinal para indicar ativo em nível baixo), o mestre envia os *bits* de dados para o dispositivo. Os dados são transmitidos em pacotes de 8 *bits* (ou mais, dependendo da configuração). Em algumas configurações do SPI, os *bits* de endereço podem ser incluídos antes dos dados para selecionar um registro ou função específica no dispositivo. No entanto, isso não é uma parte padrão do protocolo SPI e depende da implementação do dispositivo. A transmissão termina quando o mestre desativa o pino /CS (geralmente colocando-o em nível alto), indicando o fim da comunicação com o dispositivo.



(a) Transação de Escrita (b) Transação de Leitura

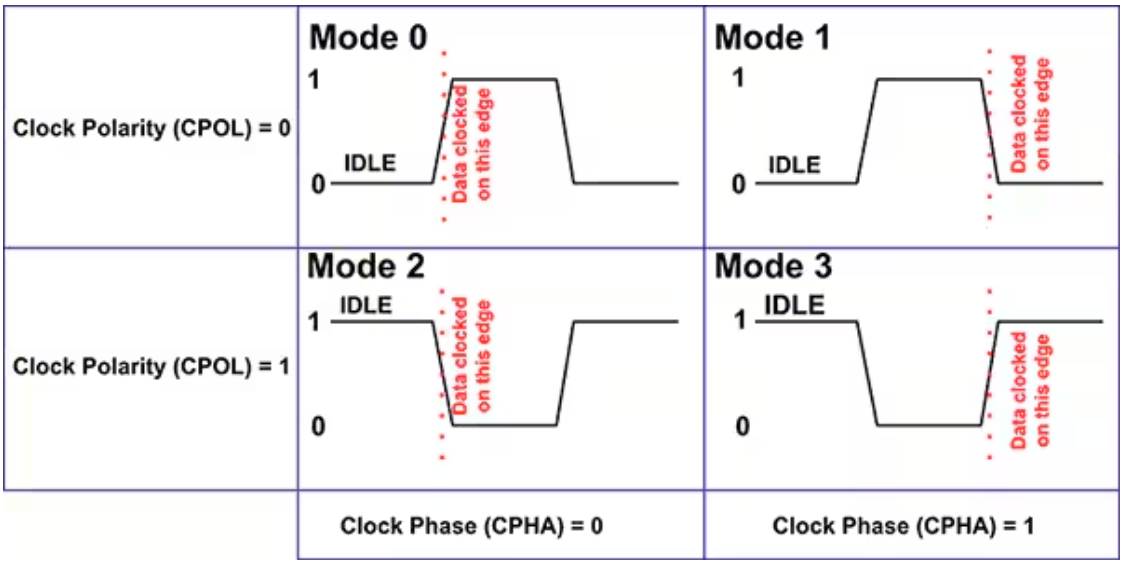
Em um barramento SPI, existe um mestre pré-definido, e um ou mais escravos, com a comunicação ocorrendo de forma **ponto-a-ponto** entre o mestre e cada escravo. O mestre sempre gera os sinais de /CS, SCLK e MOSI, e lê o sinal MISO. A seleção do escravo com quem o mestre troca informações é feita pelo sinal /CS. O mestre precisa produzir um sinal /CS distinto para cada escravo, e acionar apenas o sinal /CS correspondente ao escravo com quem se deseja comunicar. Assim, o barramento SPI usa 3 + *n* fios, onde *n* é o número de escravos. Em algumas situações, alguns dos fios podem ser omitidos. Por exemplo, se o mestre apenas envia dados a todos os escravos, sem receber nada, pode-se omitir a linha MISO. Se o mestre apenas recebe dados dos escravos, pode-se omitir a linha MOSI. Se o barramento possui apenas um escravo e este não necessita das transições de /CS para controlar as transferências, pode-se omitir o sinal /CS.

O primeiro parâmetro importante da SPI é a **velocidade** de transmissão, em *bits* por segundo, Esta é definida no mestre, não precisando ser pré-definida nos escravos, pois os mesmos recebem o *clock* gerado pelo mestre. O segundo parâmetro é a **ordem dos *bits***, que precisa ser estabelecida previamente entre mestre e escravo. O mais comum é enviar o *bit* mais significativo primeiro (*MSB first*).

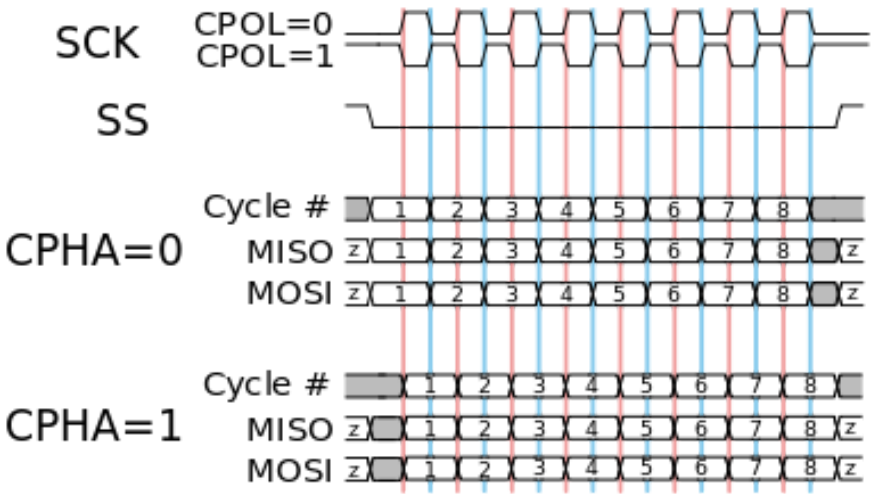
Outro parâmetro, também pré-estabelecido entre mestre e escravos, é o **formato**. Os formatos ***Motorola*** e ***TI*** (Texas Instruments) diferem principalmente na forma como a sincronização do *clock* e dos dados é feita, influenciando o momento em que os dados são capturados e enviados. O formato **Motorola** é o mais comum no padrão SPI. Esse formato é configurável usando dois parâmetros principais, a **fase** e a **polaridade** do *clock*. Falaremos destes dois parâmetros adicionais mais adiante.

O formato **TI** tem uma diferença crucial: ele inclui um pulso adicional no início de cada transmissão, conhecido como ***start pulse***. Esse pulso é enviado no início de cada transferência SPI para indicar claramente o início de uma transmissão de dados. Em termos práticos, o pulso de início ocorre imediatamente antes de começar a transmissão de dados. O /CS não é mantido ativo durante toda a transmissão; ele pode ser liberado enquanto o pulso de início ocorre. O dado é sempre transmitido no primeiro ciclo de *clock* logo após o pulso de início. Essa adição torna o formato TI menos comum do que o Motorola, mas é útil em alguns sistemas específicos da Texas Instruments para garantir a sincronização precisa do início da transferência de dados.

O último parâmetro importante, específico do formato Motorola, e também pré-estabelecido, é o **modo**, composto de dois parâmetros. A polaridade do *clock* (*bit* CPOL) determina o estado ocioso do sinal do clock (alto ou baixo). Ou seja, se CPOL = 0, o *clock* permanece em nível baixo entre transações SPI. A **fase** do clock (*bit* CPHA) determina quando os dados são capturados em relação ao ciclo do *clock*. Se CPHA = 0, a captura dos dados ocorre na **primeira borda**, enquanto se CPHA = 1, a captura ocorre na **segunda borda**. Note que o modo não define borda de subida ou descida. Se CPOL = 0, a primeira borda será de subida e se CPOL = 1, a primeira borda será de descida. Esses dois parâmetros permitem quatro configurações diferentes (modos SPI 0, 1, 2, 3), dependendo de como CPOL e CPHA são configurados (o número é definido em binário, com os *bits* na ordem CPOL,CPHA).



Fonte: DigiKey



Fonte: Typhoon HIL Documentation

Note que o protocolo SPI define apenas a forma de transferência de *bytes*. O que cada dispositivo faz com os *bytes* recebidos e o formato dos dados nos *bytes* transmitidos depende das especificações do dispositivo. Por isso, é sempre importante estudar o *datasheet* do dispositivo específico usado no projeto.

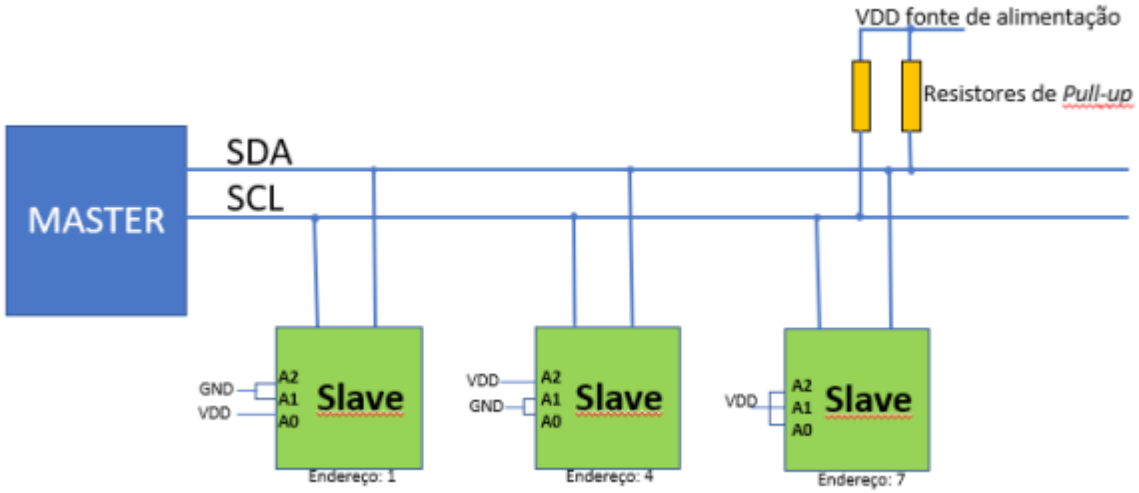
Interface serial I2C

O barramento I2C foi inventado pela Philips (NXP) em 1982. É um padrão mundial *de facto* que hoje em dia é implementado em, aproximadamente, 1000 diferentes CI's fabricados por mais de 50 empresas. Adicionalmente, o barramento I2C inspirou várias arquiteturas de controle como *System Management Bus* (SMBus), *Power Management Bus* (PMBus), *Intelligent Platform Management Interface* (IPMI), *Display Data Channel* (DDC) e *Advanced Telecom Computing Architecture* (ATCA). O barramento I2C está presente na maioria dos sistemas embarcados.

O barramento I2C tem somente dois fios bidirecionais em *open collector* ou *open drain*, sendo uma linha de dados (SDA) e uma linha de relógio (SCL). Este é um sistema de comunicação multimestre, permitindo que múltiplos mestres (*Master*) envie dados para vários escravos (*Slaves*). Um dispositivo pode atuar tanto como mestre quanto como escravo. Isso significa que um único dispositivo pode iniciar a comunicação e controlar o barramento (modo mestre) em um momento, e em outro momento, pode responder a solicitações de um outro dispositivo mestre (modo escravo). É importante lembrar que a capacidade de um dispositivo ser tanto mestre quanto escravo é uma capacidade do I2C, não implicando que todos os dispositivos I2C usem esta capacidade. Muitos dispositivos I2C (como os que veremos nos projetos) funcionam exclusivamente como escravo, e em muitos sistemas embarcados o microcontrolador atua exclusivamente como mestre.

Abaixo ilustra um barramento I2C com um mestre (*Master*) que pode enviar dados para vários escravos (*Slaves*). Cada escravo tem seu próprio endereço (*device address*),

pré-definido pelo fabricante e previamente conhecido pelo mestre. Muitas vezes os *bits* menos significativos do dispositivo podem ser configurados pelos níveis lógicos de alguns pinos. No exemplo, os três *bits* menos significativos são configurados pelas entradas A0-A2 Na medida que o mestre precisa enviar uma informação para um escravo, ele endereça o seu alvo, envia os dados e termina a transferência. O protocolo I2C é *half-duplex*, pois um mestre pode tanto enviar como receber dados para/do escravo, porém não pode fazer ambos ao mesmo tempo.



Os CIs compatíveis com o barramento I2C permitem que um projeto de sistema progrida rapidamente diretamente do diagrama de blocos funcional para um protótipo. Além disso, como eles são conectados diretamente no barramento I2C sem nenhuma interface externa adicional, eles permitem que um sistema de protótipo seja modificado ou atualizado simplesmente “ligando” ou “soltando” CIs do, ou para o, barramento.

Abaixo estão listados alguns recursos dos CIs compatíveis com o barramento I2C que são particularmente atraentes para os projetistas:

- Os blocos funcionais no diagrama de blocos correspondem aos CIs reais; os projetos avançam rapidamente do diagrama de blocos para o esquema final.
- Não há necessidade de projetar interfaces de barramento, porque a interface de barramento I2C já está integrada no *chip*.
- O protocolo integrado de endereçamento e transferência de dados permite que os sistemas sejam completamente definidos por *software*.
- Os mesmos tipos de CI costumam ser usados em muitas aplicações diferentes.
- O tempo de projeto reduz à medida que os projetistas se familiarizam rapidamente com os blocos funcionais frequentemente representados pelos CI’s compatíveis com barramento I2C.
- Os CIs podem ser adicionados ou removidos de um sistema sem afetar outros circuitos no barramento.
- O diagnóstico e a depuração de falhas são simples.

- O tempo de desenvolvimento de *software* pode ser reduzido montando uma biblioteca de módulos de *software* reutilizáveis.

Além dessas vantagens, os CIs CMOS compatíveis com o barramento I2C oferecem aos projetistas recursos especiais que são particularmente atraentes para equipamentos portáteis e sistemas alimentados por bateria. Todos eles têm:

- Consumo de corrente extremamente baixo,
- Alta imunidade a ruídos,
- Ampla faixa de tensões de alimentação,
- Ampla faixa de temperaturas de operação.

O barramento I2C é popular porque é simples de usar. Ao contrário do SPI, que é menos comum atualmente, o I2C permite a existência de mais de um mestre. Apenas a velocidade superior do barramento é definida, e apenas dois fios com resistores *pull-up* são necessários para conectar um grande número (até 127) de dispositivos I2C. O I2C pode usar microcontroladores lentos com pinos GPIO, pois eles só precisam gerar condições corretas de Início e Fim (*Start / Stop*), além de funções para ler e escrever um *byte*.

A transferência de e para o dispositivo mestre é serial e é dividida em pacotes de 8 *bits*. Todos esses requisitos simples simplificam a implementação da interface I2C, mesmo com microcontroladores baratos que não possuem controlador de *hardware* I2C específico. Só são necessários 2 pinos de E/S livres e que possam ser configurados como *open collector / open drain*, e poucas rotinas I2C simples para enviar e receber comandos. As especificações I2C iniciais definiam a frequência máxima do *clock* em 100 kHz. Mais tarde, isso foi aumentado para 400 kHz no ***Fast Mode***. Há ainda um ***High Speed Mode*** que pode ir até 3,4 MHz e também um ***Ultra-Fast Mode*** de 5 MHz.

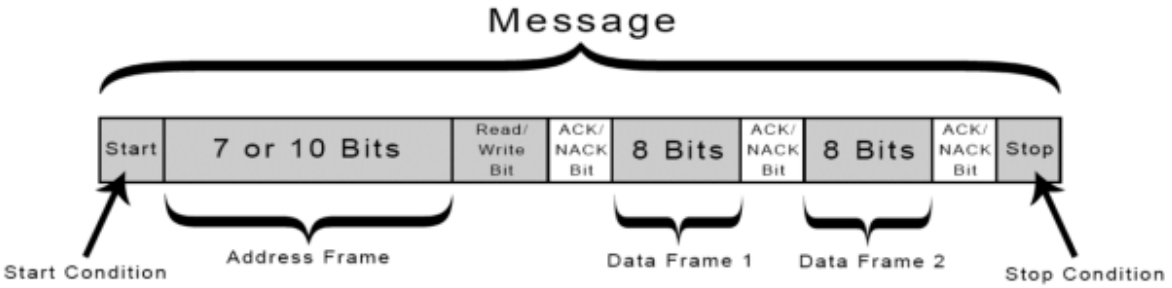
Endereços I2C

A comunicação I2C básica usa transferências de 8 *bits* (1 *byte*). Cada **dispositivo escravo** I2C **possui um endereço** de 7 *bits* (*device address*), que precisa ser exclusivo num barramento. Alguns dispositivos têm endereço I2C fixo, enquanto outros têm algumas entradas que determinam os *bits* menos significativos do endereço I2C. Isso torna muito fácil ter todos os dispositivos I2C no barramento com endereços exclusivos. Também existe um modo I2C com endereços de 10 *bits*. No endereçamento de 7 *bits*, os *bits* 7 a 1 representam o endereço, enquanto o *bit* 0 é usado para sinalizar a operação (leitura ou escrita) no dispositivo. Se o *bit* 0 (no *byte* do endereço) estiver definido como 1, o dispositivo mestre lerá o dispositivo I2C escravo.

O **dispositivo mestre não precisa de endereço**, pois é ele que gera o relógio (via SCL) e endereça individualmente os dispositivos escravos I2C. Ou seja, o mestre é o dispositivo que está gerenciando a transferência de informação.

Protocolo I2C

No estado normal (*idle*), ambas as linhas (SCL e SDA) estão em nível alto. A comunicação é iniciada pelo dispositivo mestre através do sinal de **START**. É importante notar que qualquer dispositivo no barramento pode se tornar o mestre durante aquele ciclo de transferência (barramento *multi-master*), bastando produzir o sinal de START. No I2C, os dados são transferidos em mensagens. As mensagens são divididas em quadros de dados. Cada mensagem possui um quadro de endereços, que contém o endereço binário do escravo, e um ou mais quadros de dados, que contêm os dados que estão sendo transferidos. A mensagem também inclui condições de início e parada, *bits* de leitura/gravação e bits **ACK/NACK** (*Acknowledge / Non-Acknowledge*) entre cada quadro de dados.



Passos da transmissão de dados no protocolo I2C

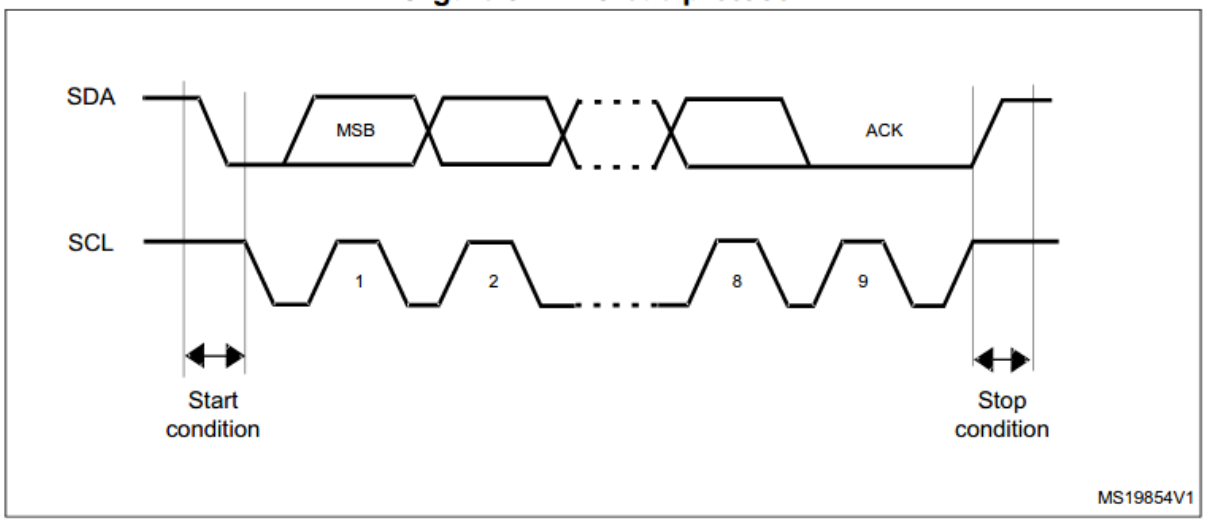
Inicialmente, para entender a transmissão de dados no protocolo I2C, é importante considerar a restrição na alteração do nível lógico do sinal SDA. As transições de nível lógico em SDA devem ocorrer apenas enquanto o sinal SCL está em nível baixo, com apenas duas exceções. O sinal de *START* (início do ciclo de transmissão) é executado realizando a mudança no SDA do nível alto para o baixo com SCL em nível alto. O sinal de *STOP* (final do ciclo de transmissão) é executado realizando a mudança no SDA do nível baixo para o alto com SCL em nível alto.

1. O mestre envia uma condição de *START* para todos os escravos conectados. Ao executar o *START*, o mestre sinaliza que ele controla o barramento durante aquele ciclo.
2. O mestre envia aos escravos conectados os 7 ou 10 *bits* de endereço do escravo com o qual ele quer se comunicar, junto com o *bit* de leitura/escrita. O mestre muda o SDA, se necessário, enquanto SCL está em nível baixo e os escravos leem o valor em SDA na borda de subida do SCL. Assim, o mestre executa 8 (ou 11) ciclos de *clock* em SCL. Na sequência, o mestre executa ainda mais um ciclo de *clock* adicional.
3. Cada escravo compara o endereço enviado pelo mestre com seu próprio endereço. Se o endereço for o seu, o escravo retorna um *bit* de *ACK* (*Acknowledge*), fazendo que a linha SDA fique em nível baixo durante o ciclo adicional de *clock* gerado pelo mestre. Se o endereço recebido não coincidir com o seu endereço, o escravo deixa a linha SDA em nível alto.
4. Se nenhum dos escravos do barramento tiver seu endereço correspondendo ao endereço enviado pelo mestre, nenhum deles irá produzir o sinal de *ACK*. Desta forma, a linha SDA permanece em nível lógico alto durante o ciclo adicional de *clock*,

e o mestre entende esta condição como dispositivo escravo ausente do barramento (*NACK*). Assim, o mestre aborta o ciclo gerando um sinal de *STOP* e liberando o barramento (linhas SDA e SCL em nível alto).

5. O mestre envia ou recebe o quadro de dados, controlando o sinal SCL por nove ciclos. Os *bits* de dados são transferidos nos oito primeiros ciclos.
6. Após cada quadro de dados ser transferido, o dispositivo que o recebeu retorna um *bit* de *ACK* para quem o enviou, para reconhecer a recepção com sucesso do quadro. Se a operação é de leitura (o mestre recebe), quando o mestre recebe o último *byte* que deseja receber, ele retorna um *bit* de *NACK* para indicar que não deseja mais receber *bytes*.
7. Para parar a transmissão de dados, o mestre envia uma condição de STOP, liberando o barramento.

Figure 512. I²C bus protocol



Há ainda mais detalhes sobre o protocolo. Um deles é o sinal de *REPEATED START*. Este consiste de um novo sinal de *START* sem que tenha ocorrido o sinal de *STOP*. Este sinal é muito usado quando se deseja passar de uma operação de escrita para uma de leitura (ou vice-versa) sem que o Mestre libere o barramento. Sem o sinal de *STOP*, o Mestre continua comandando o barramento e o novo *START* sinaliza que a comunicação continua, mas o tipo de operação (leitura ou escrita) pode mudar. Por exemplo, alguns dispositivos periféricos I2C possuem vários registradores internos. Para ler um deles, o Mestre deve inicialmente executar uma operação de escrita enviando o *byte* (ou *bytes*) que especificam o endereço interno do registrador. A seguir, o Mestre deve iniciar uma operação de leitura para obter os dados desejados.

No protocolo I2C, ACK (*Acknowledgment*) e NACK (*Not Acknowledgment*) são sinais enviados entre dispositivos para indicar a conclusão de uma operação de comunicação.

- ACK (*Acknowledgment*): Quando um dispositivo recebe um *byte* de dados, ele envia um sinal ACK para indicar que recebeu os dados com sucesso e que está pronto para

receber mais. Esse sinal é crucial para a continuidade da comunicação, pois permite que o transmissor saiba que o receptor está pronto para o próximo byte.

- NACK (*Not Acknowledgment*): O NACK é enviado pelo receptor para indicar que ele não recebeu os dados corretamente ou que não está pronto para receber mais *bytes*. Isso pode ocorrer, por exemplo, quando o receptor não pode processar mais dados ou quando o transmissor chega ao final da transmissão e precisa encerrar a comunicação. Esses sinais são fundamentais para garantir a integridade da comunicação e o controle do fluxo de dados entre os dispositivos I2C.

Existe ainda a possibilidade de arbitragem automática do barramento quando dois dispositivos tentam se tornar Mestres do barramento simultaneamente, mas o processo é relativamente complexo, fugindo do escopo deste texto.

Note que, da mesma forma que no protocolo SPI, o I2C define apenas a forma de transferência de *bytes* no barramento. A interpretação dos bytes transferidos depende do protocolo específico implementado pelo fabricante do dispositivo. Cada dispositivo I2C pode ter comandos, registradores e formatos de dados únicos que devem ser consultados no datasheet. O *datasheet* do dispositivo específico sempre deve ser consultado.

Exemplo: Comunicação I2C com memória EEPROM 24CXX

A família de memórias *EEPROM* **24CXX** consiste em dispositivos de armazenamento não volátil que utilizam o padrão I2C (*Inter-Integrated Circuit*) para comunicação. São amplamente usadas em projetos eletrônicos para armazenar dados de forma persistente, mesmo quando não há fornecimento de energia. A sigla "XX" no nome refere-se à capacidade de armazenamento de cada membro da família (dada em *kilobits*, e não em *kilobytes*), como 24C02 (2 *Kbits*), 24C16 (16 *Kbits*), e assim por diante.

Características principais:

1. Capacidades: Variam de 2 *Kbits* (24C02) até 512 K*bits* (24C512) ou mais, organizados em diferentes números de endereços e páginas.
2. Interface de Comunicação: Utilizam o protocolo I2C, permitindo a comunicação simples e de baixa pinagem (usando apenas dois pinos: SDA e SCL).
3. Tensão de Operação: Geralmente de 1.7V a 5.5V, tornando-as compatíveis com diversos microcontroladores, como Arduino, ESP32, STM32, etc.
4. Endereçamento: Cada dispositivo pode ser configurado para ter um endereço único no barramento I2C, permitindo o uso de vários *chips* na mesma linha de comunicação (tipicamente até 8).
5. Ciclos de Gravação: Suportam tipicamente 1 milhão de ciclos de gravação/apagamento por célula de memória.
6. Retenção de Dados: Os dados armazenados são retidos por até 100 anos.
7. Tamanho das Páginas: A maioria das memórias dessa família permite a escrita de dados em blocos (páginas), com o tamanho de cada página variando entre 8 a 128 *bytes*, dependendo do modelo.

A simplicidade de uso e a compatibilidade com o barramento I2C tornam a família 24CXX uma escolha popular em sistemas embarcados e projetos de *hobby* com microcontroladores.

Todos os modelos, com suas diversas capacidades de memória, apresentam um encapsulamento com a mesma pinagem, sendo dois pinos para alimentação (Vcc e GND), dois para a interface (SCL e SDA), um para proteção contra gravação (WP, deve ser mantido em nível baixo para permitir a gravação), e três para definir o *device address* (A_2, A_1 e A_0). Os níveis lógicos destes três pinos definem os três *bits* menos significativos do *device address* (pino em nível baixo faz com que o *bit* correspondente no *device address* seja 0).

Em alguns modelos, os três *bits* (ou alguns deles) de definição do *device address* são redirecionados para a definição de parte do endereço interno da EEPROM. Isto é feito para evitar a necessidade de se enviar 2 *bytes* de dados para definir o endereço interno. Vamos considerar como exemplo inicial o 24C02, que possui 2*Kbits* de espaço, correspondente a 256 *Bytes*. Para endereçar internamente a EEPROM, basta um *byte*. Assim, em uma operação de escrita, o Mestre envia a seguinte sequência de sinais e *bytes*:

START - $A_6A_5A_4A_3A_2A_1A_0$0 - $E_7E_6E_5E_4E_3E_2E_1E_0$ - Dado1 - Dado2 - … - Dadon - STOP,

onde A_x é o *bit x* do *device address* (com $A_{2\text{-}0}$ definidos nos pinos externos), o *bit* de *Read / Write* é 0, E_x é o *bit x* do endereço interno da EEPROM onde a escrita deve iniciar, e Dado*x* são os *bytes* de dados a serem escritos. O *Byte* de *device address* vai ter a sequência binária $1010A_2A_1A_0$0.

Na operação de leitura na EEPROM, o endereço interno não é transmitido ao componente. A EEPROM retorna o dado presente no endereço indicado por um **ponteiro interno**, cujo valor é carregado com os *bits* E_x. Assim, para se ler a partir de um endereço desejado, deve-se iniciar uma operação de escrita passando o *byte* de endereço e logo depois executar um REPEATED START (idêntico ao START, porém ocorrendo enquanto o barramento ainda está ocupado pelo mestre), mudando a operação para leitura. A sequência fica:

START - $A_6A_5A_4A_3A_2A_1A_0$0 - $E_7E_6E_5E_4E_3E_2E_1E_0$ - REPEATED START - $A_6A_5A_4A_3A_2A_1A_0$1 - Dado1 - Dado2 - … - Dadon - STOP (o Master responde cada Dado com um ACK, exceto no último dado, quando responde com um NACK).

Se for usado um 24C04, teremos 512 *Bytes*. Assim, precisamos de 9 *bits* para definir o endereço interno da EEPROM. Seriam necessários 2 *Bytes* de endereço interno para adicionar apenas um *bit*, o que é um desperdício. Neste caso, o *bit* A_0 passa a ter a função de E_8 e o pino A_0 no encapsulamento fica sem função. Ou seja, usa-se parte do *device address* para passar o *bit* mais significativo do endereço interno. Neste caso, o componente irá responder a *device addresses* da seguinte forma: $1010A_2A_1X0$, onde *X* é um *bit don't care*.

Usando um 24C08, A_1 e A_0 passam a ter a função de E_9 e E_8 respectivamente. Em um 24C16 (o componente presente na placa auxiliar), A_2, A_1 e A_0 passam a ter a função de E_{10}, E_9 e E_8, e os pinos A_2, A_1 e A_0 ficam sem função. Para um 24C32, o endereço interno passa a ser

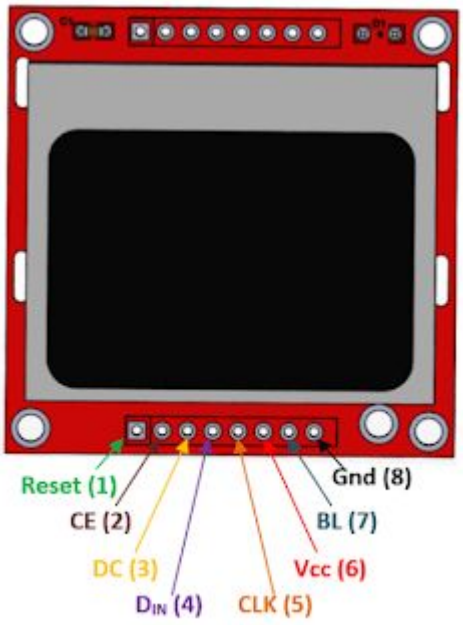
transmitido em 2 *Bytes* e os pinos de definição do *device address* voltam a executar sua função original.

É importante ter em mente que operações de escrita só podem acontecer dentro de uma mesma página da EEPROM. Se na transmissão os dados passarem do limite de uma página para outra, os dados serão escritos de forma errada dentro da mesma página. No caso da 24C16, o tamanho de página é de 16 *Bytes*. Assim, não se pode escrever uma sequência de dados que passe do endereço 15 para o 16, do 31 para o 32, etc. Para a leitura sequencial, não há restrições.

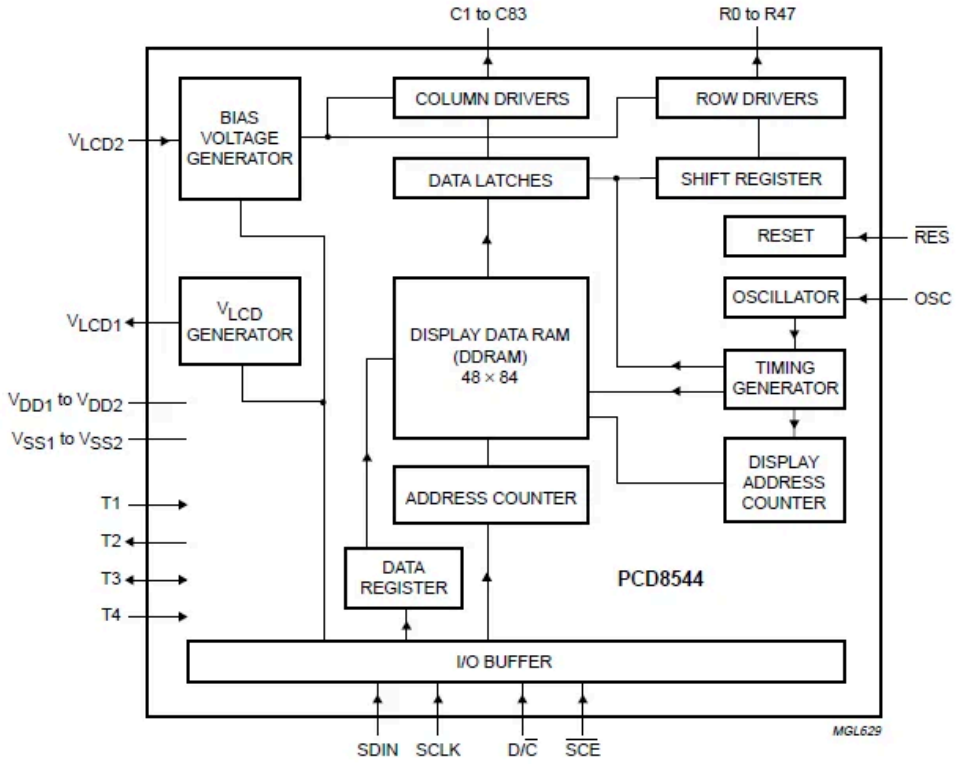
EXEMPLOS DE SENSOR E ATUADOR PROGRAMÁVEIS

Nesta seção, vamos explorar dois exemplos práticos de sensores e atuadores programáveis, destacando como as interfaces seriais síncronas, como SPI e I2C, facilitam a configuração e o controle desses dispositivos. Os sensores e atuadores escolhidos não apenas ilustram a versatilidade das interfaces seriais, mas também mostram como a integração desses componentes pode resultar em sistemas mais responsivos e precisos.

Display NOKIA 5110



O *display* **Nokia 5110** é um módulo gráfico de baixa potência, originalmente usado em celulares Nokia, que se tornou popular em projetos com Arduino e outros microcontroladores. Ele possui uma tela LCD de 84x48 *pixels*, capaz de exibir gráficos e textos em formato monocromático (preto e branco). O controlador integrado é o **PCD8544**, que se comunica com o microcontrolador via interface SPI.



As principais características deste *display* são:

- Resolução: 84x48 *pixels*.
- Controle: feito pelo controlador PCD8544.
- Interface: SPI (Serial Peripheral Interface).
- Tensão de operação: 3.3V (pode precisar de adaptadores de nível lógico para uso com placas Arduino de 5V).
- Baixo consumo de energia: 200 a 300 μA quando ativo e 1μA em modo *sleep*, ideal para projetos com baterias.
- Tamanho compacto: aproximadamente 45mm x 45mm.

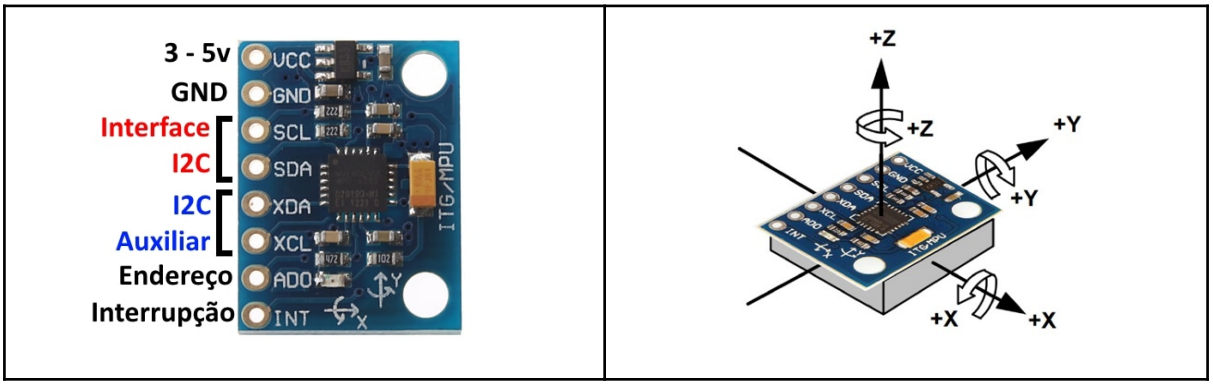
O *display* é amplamente utilizado em projetos que precisam de uma interface gráfica simples e de baixo custo, como monitores de sensores, relógios e pequenos jogos. Normalmente ele é vendido na forma de um módulo com o *display* montado e os pinos de seu controlador disponibilizados em uma fileira de pinos. Além disso, o módulo conta com uma luz de fundo (*backlight*) por LEDs, controlada por um dos pinos do módulo. Os LEDs são controlados por um transistor *drive*, portanto uma saída GPIO do microcontrolador pode controlar diretamente o acionamento do *backlight*, mantendo uma corrente de acionamento no pino muito baixa.

Além da interface padrão SPI (unidirecional *slave*, sem o sinal MISO), o módulo possui mais três pinos para sua configuração: um pino de RESET (ativo em nível baixo), um pino de *Data / Command* (**D/C**) e um pino para acender o *backlight* (ativo em nível alto ou baixo, dependendo do fabricante do módulo). O pino D/C determina se o *byte* recebido é um comando (nível baixo) ou um dado (nível alto). Para os comandos existe uma tabela

específica. Para os dados, cada *bit* dentro do *byte* especifica um *pixel* do *display* (1 para ativo, 0 para inativo). A posição do *pixel* depende da configuração realizada com os *bytes* de comando, mas geralmente cada *byte* especifica uma coluna vertical de 8 *pixels*, com o *bit* menos significativo correspondente ao *pixel* superior. Para escrever texto neste *display*, as letras precisam ser "desenhadas", ou seja, deve-se estabelecer os *pixels* a serem ativados de acordo com a letra e sua posição.

O *display* conta com alguns recursos, como ajuste de contraste, rotação de tela a cada 90 graus e de inversão total da tela, tornando os *pixels* ativos transparentes e os inativos pretos (o oposto do modo normal). Além disso, apresenta um modo *sleep*, onde a tela é desligada mas a interface SPI e a memória com os valores dos *pixels* são mantidos ativos.

Sensor de movimentos MPU6050



O MPU6050 é um sensor inercial muito popular, amplamente utilizado em sistemas de controle e navegação, dispositivos móveis, robótica e projetos de eletrônica. Ele integra dois sensores principais: um acelerômetro de 3 eixos e um giroscópio de 3 eixos, capazes de medir a aceleração linear e a velocidade angular em torno dos eixos X, Y e Z, respectivamente. Isso permite uma medição precisa de movimentos e inclinações em várias direções, tornando-o ideal para aplicações que exigem detecção de movimento em tempo real.

O MPU6050 é um sensor que combina um acelerômetro e um giroscópio de 3 eixos, oferecendo uma ampla gama de funcionalidades para medir movimentos e orientações. O acelerômetro, com fundos de escala configuráveis que variam de ±2g a ±16g onde "g" é a aceleração da gravidade terrestre, aproximadamente 9,81 m/s², é capaz de capturar a aceleração linear em cada eixo, fornecendo dados valiosos sobre inclinação, gravidade e vibrações. A escolha da faixa de fundo de escala define quantos "contagens" correspondem a uma aceleração de 1g. Para converter os valores obtidos do MPU6050, é necessário considerar os valores de fundo de escala configurados para o acelerômetro e para o giroscópio. Esses valores de fundo de escala determinam a sensibilidade do sensor e são especificados em unidades de aceleração e velocidade angular.

A fórmula para converter o valor bruto de 16 bits do acelerômetro (***Valor_Bruto_Acc***) em aceleração física (***a***) em metros por segundo ao quadrado (m/s²) é:

$$a = \frac{Valor\_Bruto\_Acc}{2^{15}} \times FS\_Acc \times 9.8\frac{m}{s^2}$$

Onde:

- ***Valor_Bruto_Acc*** é o valor obtido diretamente do acelerômetro (varia de -32768 a +32767).
- 2^{15} é o valor máximo positivo de 16 bits (magnitude = 32768).
- ***FS_Acc*** é o fundo de escala do acelerômetro em "***g***" (2, 4, 8 ou 16).
- 9,8m/s^2 é a aceleração da gravidade (1***g***).

Se, por exemplo, o fundo de escala for ±2g, temos ***FS_Acc*** = 2. A fórmula ficaria assim:

$$a = \frac{Valor\_Bruto\_Acc}{2^{15}} \times 2 \times 9.8\frac{m}{s^2}$$

Complementando essa funcionalidade, o giroscópio também opera em três eixos e pode medir a velocidade angular com escalas que vão de ±250 a ±2000 graus por segundo.

O giroscópio do MPU6050 mede a velocidade angular nos eixos X, Y e Z, e também retorna valores numéricos de 16 bits com sinal em complemento de 2. Esses valores devem ser convertidos para unidades de graus por segundo (°/s). O fundo de escala do giroscópio pode ser configurado nas seguintes faixas: ±250°/s, ±500°/s, ±1000°/s e ±2000°/s

O fundo de escala escolhido define quantos "contagens" correspondem a uma rotação de 1°/s.

A fórmula para converter o valor bruto de 16 bits do giroscópio ***Valor_Bruto_Gyro***) em velocidade angular (ω) em graus por segundo (°/s) é:

$$\omega = \frac{Valor\_Bruto\_Gyro}{2^{15}} \times FS\_Gyro$$

Onde:

- ***Valor_Bruto_Gyro*** é o valor obtido diretamente do giroscópio(varia de -32768 a +32767).
- 2^{15} é o valor máximo positivo de 16 bits (magnitude = 32768).
- ***FS_Acc*** é o fundo de escala do acelerômetro em "***g***" (250, 500, 1000, 2000).

Por exemplo, para um fundo de escala de ±500°/s (***FS_Gyro*** = 500), a fórmula seria:

$$\omega = \frac{Valor\_Bruto\_Gyro}{2^{15}} \times 500^{o}/s$$

Exemplos:

a) Se o valor bruto do acelerômetro no eixo X for 16384 e o fundo de escala configurado for ±4g, a aceleração correspondente seria:

$$a = \frac{16384}{32768} \times 4 \times 9.8\frac{m}{s^2} = 19,62\frac{m}{s^2}$$

b) Se o valor bruto do giroscópio no eixo Z for -16384 e o fundo de escala configurado for ±1000°/s, a velocidade angular correspondente seria:

$$\omega = \frac{-16384}{32768} \times 1000^{o}/s = -500^{o}/s$$

Outro aspecto do MPU6050 é o seu *Digital Motion Processor* (DMP) integrado, que permite o processamento interno dos dados provenientes do acelerômetro e do giroscópio. Esse processador é capaz de realizar cálculos complexos, como a fusão de sensores, utilizando algoritmos avançados, como o filtro de Kalman ou o filtro complementar. Isso resulta em medições mais precisas dos ângulos de inclinação e orientação, como *roll*, *pitch* e *yaw*, sem sobrecarregar o microcontrolador responsável pela operação do sistema. Além disso, o MPU6050 inclui filtros digitais passa-baixa (DLPF) configuráveis, que suavizam ruídos em ambientes de alta vibração ou interferência, melhorando ainda mais a precisão das medições. O sensor também possui um sensor de temperatura embutido, que pode ser utilizado para compensar variações térmicas que possam influenciar as medições, ou simplesmente para monitoramento térmico, garantindo uma performance otimizada em diversas condições operacionais.

O MPU6050 se destaca pela sua taxa de amostragem, que pode alcançar até 1 kHz, ou seja, ele é capaz de realizar até 1000 medições por segundo. Essa característica garante respostas rápidas e precisas, o que é fundamental em aplicações que exigem alta performance, como monitoramento de movimento e orientação. Além disso, o sensor utiliza uma interface de comunicação I2C, com um endereço padrão de 0x68, facilitando sua integração em plataformas populares como Arduino, Raspberry Pi e ESP32. Para aumentar a flexibilidade na comunicação, algumas versões do módulo, como o MPU6000, também suportam a interface SPI. Outro ponto positivo do MPU6050 é seu baixo consumo de energia, com uma média de apenas 3,9 mA. Isso o torna uma opção eficiente para dispositivos alimentados por bateria, como *wearables* e drones, que precisam operar por longos períodos sem recarga. Além de todas essas características, o *chip* tem um tamanho compacto de apenas 4x4 mm, facilitando sua integração em projetos de *hardware* onde o espaço é limitado.

Devido à sua versatilidade e conjunto abrangente de funcionalidades, o MPU6050 é amplamente utilizado em diversas aplicações. Uma de suas principais utilizações é na estabilização de drones e robôs, onde o sensor desempenha um papel crucial em manter a estabilidade e o equilíbrio, corrigindo rotações e inclinações em tempo real. Além disso, o MPU6050 é empregado em dispositivos de controle de gestos e movimentos, como consoles de videogame e interfaces de realidade aumentada, permitindo interações mais intuitivas e imersivas. Outra aplicação significativa é na navegação inercial, onde o sensor ajuda a rastrear a posição e a direção de um objeto em movimento, tornando-o essencial para sistemas de navegação em ambientes variados. O MPU6050 também é muito popular em plataformas de eletrônica, sendo frequentemente utilizado em projetos com Arduino, Raspberry Pi e ESP32, onde é necessária uma medição precisa de movimento. Por fim, em dispositivos *wearables* e *fitness trackers*, o MPU6050 é utilizado para rastrear atividades físicas e detectar movimentos corporais, contribuindo para um monitoramento eficaz da saúde e do desempenho físico. Essas diversas aplicações evidenciam a importância do MPU6050 em tecnologias modernas, abrangendo desde entretenimento até saúde e navegação.

STM32H7A3

O STM32H7A3 oferece uma variedade de módulos de comunicação serial, incluindo interfaces UART, I2C, SPI e CAN (do inglês, *Controller Area Network*), o que possibilita uma ampla gama de aplicações em projetos eletrônicos. No Roteiro 7, aprofundamos nossa exploração do módulo UART, enquanto neste Roteiro focaremos nas funções disponibilizadas pelos módulos SPI e I2C.

Módulo SPI

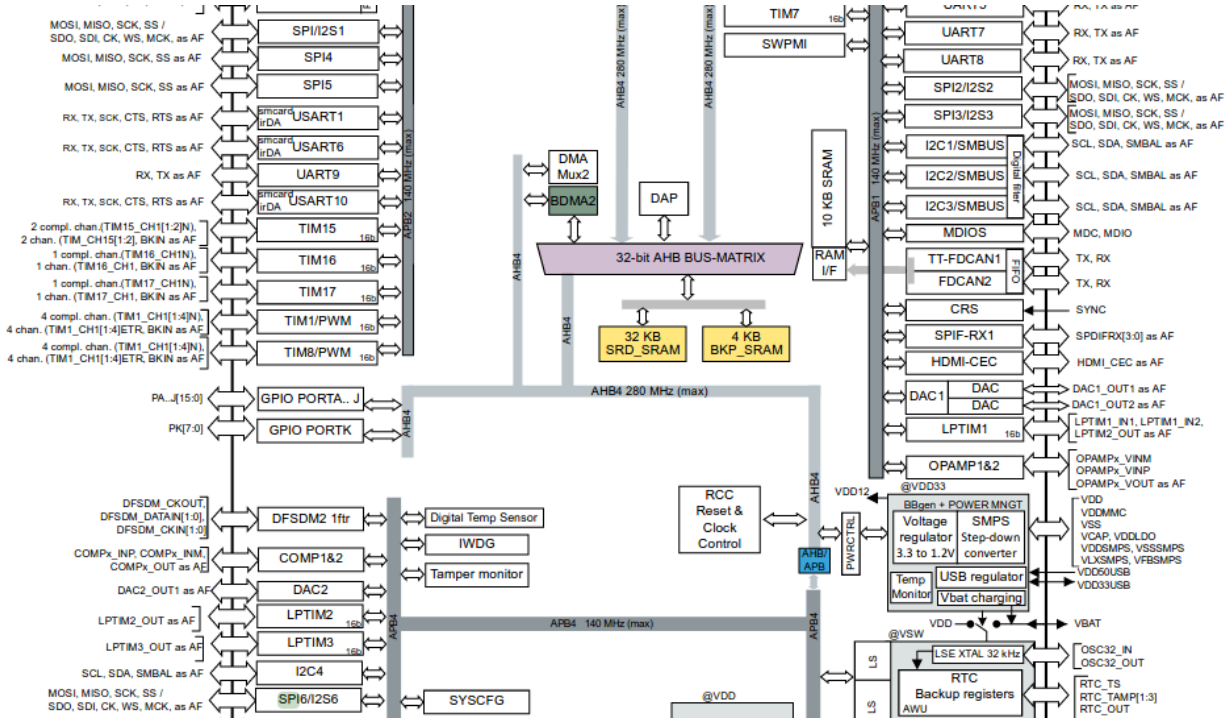
O microcontrolador STM32H7A3 é equipado com 6 módulos SPI que facilitam a comunicação serial periférica:

- SPI1, SPI2, SPI3: Esses módulos são geralmente utilizados para comunicação serial de uso geral, adequados para diversas aplicações.
- SPI4, SPI5: Oferecem flexibilidade adicional, permitindo a operação com um *clock* distinto do barramento de interface.
- SPI6: Este módulo também pode operar com um *clock* independente e apresenta um *bit* de ativação adicional, o SPI6AMEN, que controla o *clock* do DFSDM2.

Esses módulos utilizam um mecanismo conhecido como *clock gating* para gerenciar a ativação dos sinais de relógio, contribuindo para a otimização do consumo de energia.

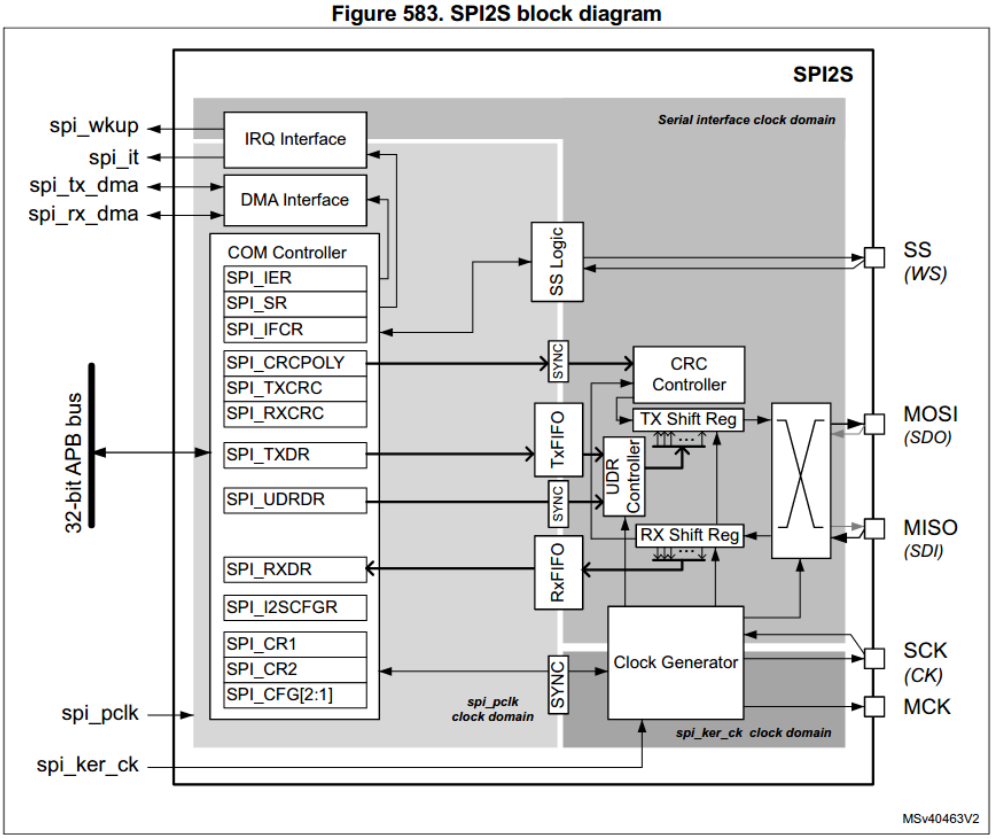
Para ativar o *clock* de um módulo SPI, é necessário configurar os *bits* `RCC_APB2ENR_SPI1EN`, `RCC_APB1LENR_SPI2EN`, `RCC_APB1LENR_SPI3EN`, `RCC_APB2ENR_SPI4EN`, `RCC_APB2ENR_SPI5EN` e `RCC_APB4ENR_SPI6EN`, conforme o módulo SPI em questão. Quando o periférico não é mais necessário, o *clock* pode ser desativado desabilitando o *bit* `SPIxEN` correspondente. Os *bits* `RCC_CDCCIP1R_SPI123SEL`, `RCC_CDCCIP1R_SPI45SEL` e

RCC_SRDCCIPR_SPI6SEL permitem escolher entre várias fontes de *clock*, como PLLs, HSI, CSI, HSE ou um *clock* externo (I2S_CKIN) para o módulo SPI correspondente. É necessário ativar o *clock* do barramento de interface AHB4 através dos *bits* GPIOxEN correspondentes ao GPIO utilizado pelo módulo SPI. Por exemplo, se o SPI1 utiliza pinos do GPIOA, o *bit* RCC_AHB4ENR_GPIOAEN deve ser habilitado.



A comunicação SPI envolve diferentes domínios de *clock*, incluindo o *clock kernel* (`spi_ker_ck`), o *clock* do barramento de interface (32-*bit* APB **bus**) e o *clock* serial do periférico (`spi_pclk`). É fundamental garantir a sincronização adequada entre esses *clocks* para evitar problemas de comunicação.

Figure 583. SPI2S block diagram

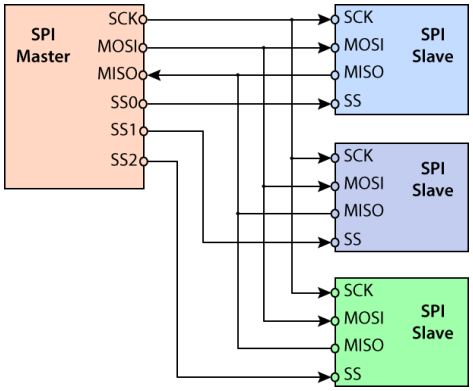


O protocolo SPI utiliza quatro sinais principais: o MOSI (*Master Out Slave In*), que é utilizado pelo mestre para enviar dados ao escravo; o MISO (*Master In Slave Out*), que permite ao escravo enviar dados de volta ao mestre; o SCK (*Serial Clock*), um sinal de *clock* gerado pelo mestre que sincroniza a transferência de dados; e o SS (*Slave Select*), que serve para selecionar qual escravo o mestre deseja comunicar.

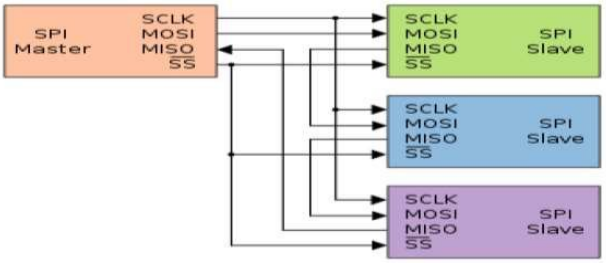
Quando a comunicação é no **modo mestre e escravo**, um único mestre se comunica com um único escravo, iniciando a comunicação e controlando o sinal de *clock* e o sinal de seleção do escravo. O escravo responde aos comandos do mestre e transfere dados conforme o *clock* fornecido. Este modo é determinado pela configuração do registrador `GPIOx_MODER` para definir os pinos de saídas para o mestre (MOSI, SCK, SS) e de entrada para o escravo (MISO) com a função alternativa (AF). Em seguida, os registradores `GPIOx_AFRL` e `GPIOx_AFRH` devem ser configurados para selecionar a função alternativa SPI correspondente aos pinos `GPIO` utilizados. No que diz respeito ao *software*, **o mestre é responsável por gerenciar o sinal SS, gerar o clock (SCK) e iniciar a transferência de dados.**

Quando se fala em **comunicação multi-escravos**, um único mestre pode se comunicar com vários escravos, cada um possuindo um sinal SS dedicado. O mestre ativa o sinal SS correspondente para selecionar qual escravo deseja comunicar. Existem duas principais topologias para essa comunicação: a **topologia em estrela**, onde cada escravo tem uma linha SS dedicada conectada a um pino GPIO, SS0, SS1 ou SS2, controlado pelo mestre, e a **topologia em cadeia Daisy (*Daisy chain*)**, na qual os escravos são conectados em série, compartilhando os sinais SCK e MOSI, com cada um tendo um sinal SS dedicado ao escravo anterior na cadeia. A topologia em estrela é simples de implementar, mas requer um número

maior de pinos GPIO para o mestre comunicar com escravos, enquanto a topologia em cadeia reduz o número de pinos GPIO necessários, mas exige um gerenciamento mais complexo do fluxo de dados.



Topologia em estrela



Topologia em cadeia *Daisy*

A configuração do SPI é realizada principalmente por meio de quatro registradores essenciais: SPI_CR1, SPI_CR2, SPI_CFG1 e SPI_CFG2. O registrador SPI_CR1 é responsável por controlar a habilitação do módulo (SPE), a polaridade do *clock* (CPOL), a fase do *clock* (CPHA) e outros parâmetros relevantes. Por sua vez, o SPI_CR2 permite definir a quantidade de dados (TSIZE) a serem transferidos em uma operação do registrador SPI_TXDR ou para o registrador SPI_RXDR e o número de transições (TSER) dessa transação. O registrador SPI_CFG1 é fundamental para definir a taxa de transmissão (MBR), o número de *bits* por quadro de dados (DSIZE), além de controlar a habilitação do código de verificação de redundância cíclica (CRCEN) e o gerenciamento de DMA (TXDMAEN e RXDMAEN). Já o SPI_CFG2 especifica se o SPI opera em modo mestre ou escravo (MASTER), se o periférico mantém o controle dos pinos GPIO associados mesmo quando o módulo SPI está desabilitado (AFCNTR) e permite a seleção do protocolo serial entre Motorola e TI (SP). Durante a configuração, **é crucial garantir que o *bit* SPI_CR1_SPE esteja resetado para 0**, assegurando que o módulo não esteja habilitado antes que todas as configurações necessárias sejam realizadas.

O *baud rate* do módulo SPI é determinado pela frequência do *clock kernel* do módulo SPI no modo mestre e o divisor configurado em SPI_CFG1_MBR:

$$baud\_rate = spi\_ker\_ck / 2^{SPI\_CFG1\_MBR+1}$$

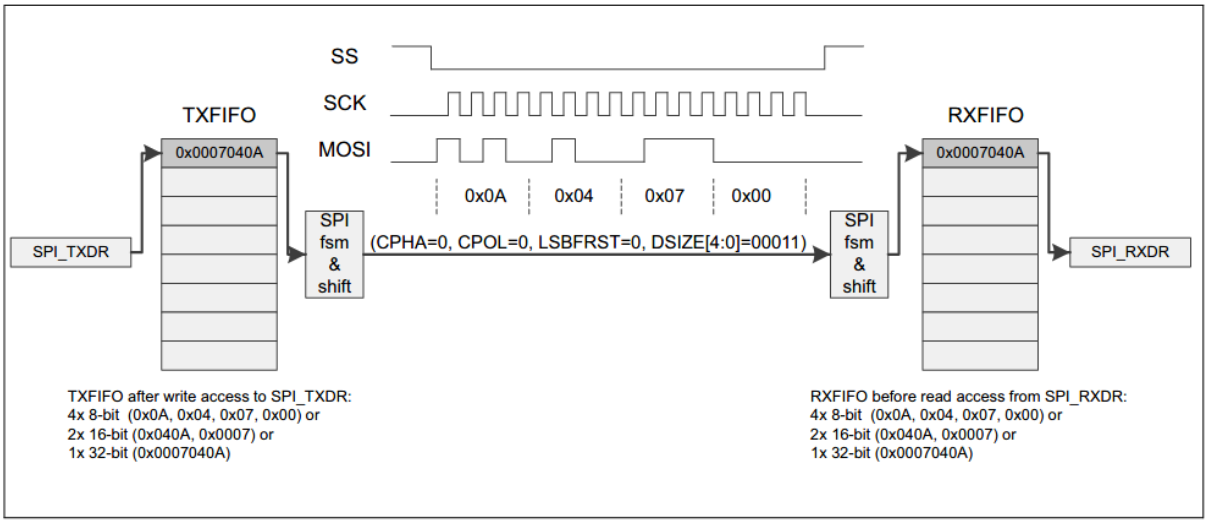
Quanto ao modo de comunicação, o SPI opera predominantemente em modo *full-duplex*, onde os dados são transmitidos e recebidos simultaneamente. Entretanto, também é possível configurar a comunicação em modo *simplex*, onde a transmissão ou recepção ocorre em apenas uma direção; isso pode ser feito através de um campo específico SPI_CFG2_COMM do registrador SPI_CFG2. Além disso, este mesmo registrador é responsável pela configuração do formato dos dados SPI, que inclui a polaridade (SPI_CFG2_CPOL) e a fase

(SPI_CFG2_CPHA) do *clock*. O modo *half-duplex*, que alterna entre transmissão e recepção, não é diretamente suportado pelo *hardware* SPI, mas pode ser implementado em *software*, gerenciando a direção da transferência através do *bit* SPI_CR1_HDDIR e os pinos de comunicação A correta configuração desses parâmetros é fundamental para assegurar a sincronização entre o mestre e o escravo durante a comunicação SPI. Se a polaridade e a fase do *clock* não forem configuradas adequadamente, pode haver amostragem ou transmissão de dados em momentos inadequados, resultando na corrupção das informações transmitidas. Portanto, uma atenção cuidadosa a esses detalhes é crucial para o sucesso da comunicação.

Para informações detalhadas sobre a configuração do SPI, consulte o Manual de Referência. As seções relevantes incluem: configuração do SPI, que é semelhante para os modos mestre e escravo (Seção 55.4.9); habilitação do SPI (Seção 55.4.10); transmissão e recepção de dados (Seção 55.4.11); e desabilitação do SPI (Seção 55.4.12).

Para otimizar a transferência de dados nos módulos SPI, tanto a estrutura de dados FIFO (do inglês, *First-In, First-Out*) quanto a DMA (*Direct Memory Access*) podem ser usados junto com o *hardware* de serialização dos *bytes* em conformidade com o protocolo SPI implementado nos módulos SPI. O módulo SPI possui duas **FIFOs internas**, uma no receptor e outra no transmissor. Essas FIFOs atuam *buffers* temporários, armazenando dados para garantir um fluxo de transferência mais contínuo. O campo SPI_CFG1_FTHLV permite configurar a quantidade de quadros de dados por pacote. É importante ressaltar a restrição quanto ao tamanho do pacote de dados, que não pode ultrapassar metade do espaço disponível na FIFO. Essa limitação é crucial para assegurar que haja espaço suficiente para acomodar os dados recebidos enquanto um pacote está sendo transmitido, prevenindo situações de *overflow* que poderiam comprometer a integridade da transferência. Além disso, a eficiência da interface SPI é otimizada quando os tamanhos dos pacotes configurados estão alinhados com o paralelismo de acesso aos registradores de dados SPI. Isso significa que, para maximizar a eficácia da transferência, o tamanho do pacote deve ser ajustado em conformidade com a largura de dados do registrador SPI. Esse alinhamento não apenas melhora a performance geral, mas também assegura um fluxo contínuo e eficiente de dados durante as operações de comunicação.

Figure 595. Packing data in FIFO for transmission and reception



1. DSIZE[3:0] is configured to 4-bit, data is right aligned, valid bits are performed only on the bus, their order depends on LSBFRST, if it is set, the order is reversed at all the data frames.

Mostramos no Roteiro 9 que o DMA é uma funcionalidade que permite a transferência de dados entre periféricos e memória, ou entre diferentes áreas da memória, sem a necessidade de intervenção da CPU. Essa abordagem libera a CPU para realizar outras tarefas durante as transferências de dados, resultando em um aumento significativo da eficiência do sistema. A configuração do DMA para operar com periféricos envolve algumas etapas cruciais. Primeiramente, é necessário habilitar o *clock* do controlador DMA desejado, utilizando os *bits* `RCC_AHB1ENR_DMA1EN` ou `RCC_AHB1ENR_DMA2EN` nos registradores. Em seguida, recomenda-se resetar o controlador DMA, acionando os *bits* `RCC_AHB1RSTR_DMA1RST` ou `RCC_AHB1RSTR_DMA2RST` no registrador `RCC_AHB1RSTR`, e o periférico SPI, setando o *bit* `SPIxRST` do registrador `RCC_APB1LRSTR`, `RCC_APB2RSTR` ou `RCC_APB4RSTR`, garantindo que os módulos estejam prontos para configuração. Uma vez realizado o *reset*, a próxima etapa é configurar o canal DMA, utilizando registradores específicos, como `DMA_SxCR`, `DMA_SxNDTR`, `DMA_SxPAR` e `DMA_SxM0AR`. Esses registradores permitem definir a direção da transferência, o número de dados a serem transferidos, além dos endereços de origem e destino, juntamente com outras opções relevantes. Deve-se, então, selecionar o canal DMAMUX que corresponde ao canal DMA configurado. Em seguida, é importante configurar o registrador de configuração do canal, o `DMAMUX_CxCR`. Nesse registrador, é necessário definir o `DMAREQ_ID`, que especifica a fonte de requisição do DMA. Para isso, é aconselhável consultar a tabela de mapeamento do DMAMUX, onde se pode identificar o código correspondente ao periférico SPI desejado, seja para transmissão (`SPIx_TX`) ou recepção (`SPIx_RX`).

Table 101. DMAMUX1: assignment of multiplexer inputs to resources (continued)

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
37	SPI1_RX	79	UART7_RX	121	Reserved
38	SPI1_TX	80	UART7_TX	122	Reserved
39	SPI2_RX	81	UART8_RX	123	Reserved
40	SPI2_TX	82	UART8_TX	124	Reserved
41	USART1_RX	83	SPI4_RX	125	Reserved
42	USART1_TX	84	SPI4_TX	127	Reserved

Table 104. DMAMUX2: assignment of multiplexer inputs to resources (continued)

DMA request MUX input	Resource	DMA request MUX input	Resource
10	LP UART1_TX	26	Reserved
11	SPI6_RX	27	Reserved
12	SPI6_TX	28	Reserved
13	I2C4_RX	29	Reserved
14	I2C4_TX	30	Reserved
15	Reserved	31	Reserved
16	Reserved	-	-

DMA request MUX input	Resource	DMA request MUX input	Resource
43	USART2_RX	85	SPI5_RX
44	USART2_TX	86	SPI5_TX
45	USART3_RX	87	SAI1_A
46	USART3_TX	88	SAI1_B
47	TIM8_CH1	89	SAI2_A
48	TIM8_CH2	90	SAI2_B
49	TIM8_CH3	91	SWPMI_RX
50	TIM8_CH4	92	SWPMI_TX
51	TIM8_UP	93	SPDIFRX_DT
52	TIM8_TRIG	94	SPDIFRX_CS
53	TIM8_COM	95	Reserved
54	Reserved	96	Reserved
55	TIM5_CH1	97	Reserved
56	TIM5_CH2	98	Reserved
57	TIM5_CH3	99	Reserved
58	TIM5_CH4	100	Reserved
59	TIM5_UP	101	DFSDM1_dma0
60	TIM5_TRIG	102	DFSDM1_dma1
61	SPI3_RX	103	DFSDM1_dma2
62	SPI3_TX	104	DFSDM1_dma3

Após realizar essas configurações, o próximo passo é habilitar o canal DMA. Isso é feito configurando o *bit* `DMA_SxCR_EN` no registrador DMA_SxCR, ativando assim o canal DMA e permitindo que ele inicie as transferências de dados conforme configurado.

O módulo SPI possui suporte para **CRC (do inglês *Cyclic Redundancy Check*)**, uma importante funcionalidade detectora de erros que aumenta a integridade dos dados transmitidos. O CRC é um código de detecção de erros que assegura a confiabilidade das informações em diversos protocolos de comunicação, incluindo o SPI. Ao transmitir dados, um valor de CRC é calculado matematicamente com base nos dados originais e anexado à transmissão. Quando os dados chegam ao receptor, este recalcula o CRC utilizando os dados recebidos e o compara com o valor de CRC que foi enviado. Se ambos os valores coincidirem, é altamente provável que os dados tenham sido transmitidos sem erros.

Para configurar o módulo SPI para utilizar o CRC, alguns registradores e *bits* específicos são utilizados. O registrador `SPI_CFG1` contém o *bit* `SPI_CFG1_CRCEN`, que deve ser habilitado para ativar o cálculo e a verificação do CRC. Além disso, o campo `SPI_CFG1_CRCSIZE` neste registrador permite definir o tamanho do polinômio CRC, que pode ser de 16 *bits* (padrão) ou 32 *bits*. O registrador `SPI_CRCPOLY` é onde se armazena o valor do polinômio CRC a ser usado, enquanto o `SPI_TXCRC` contém o valor CRC calculado para os dados que estão sendo transmitidos, e o `SPI_RXCRC` armazena o valor CRC recebido juntamente com os dados.

O procedimento para utilizar o CRC no SPI é bastante direto. Primeiro, é necessário habilitar o CRC, configurando o *bit* `SPI_CFG1_CRCEN` no registrador `SPI_CFG1`. Em seguida, o polinômio CRC desejado deve ser escrito no registrador `SPI_CRCPOLY`. Os dados podem então ser transmitidos normalmente através do registrador de dados SPI. Durante a transmissão, o módulo SPI calcula automaticamente o CRC, que é armazenado em `SPI_TXCRC`, e o anexa aos dados. Ao receber os dados, o módulo SPI também recebe o CRC, que é armazenado no registrador `SPI_RXCRC`. Por fim, o receptor recalcula o CRC com base nos dados recebidos e compara esse valor ao que está armazenado em `SPI_RXCRC`. Se os dois valores coincidirem, a transmissão é considerada íntegra; caso contrário, um erro de CRC é detectado, garantindo assim a confiabilidade da comunicação.

Por fim, o gerenciamento das interrupções é uma responsabilidade do *software*, que deve configurar, habilitar e tratar as interrupções geradas pelos periféricos SPI. Para isso, é necessário implementar rotinas de serviço de interrupção (ISRs) para as IRQs, que executarão as ações apropriadas em resposta a cada evento. Os periféricos SPI oferecem a possibilidade de habilitar interrupções para uma variedade de situações, utilizando os *bits* de controle presentes no registrador `SPI_IER`. Entre os eventos que podem gerar interrupções, destacam-se a conclusão de uma transferência de dados (`EOTIE`), a detecção de erros de comunicação, como os erros de CRC (`CRCEIE`) e erros de quadro de dados (`TIFREIE`), e eventos relacionados à FIFO. Por exemplo, interrupções podem ser geradas quando a FIFO está cheia (`OVRIE`) ou vazia (`UDRIE`), atingiu-se o limiar de TXFIFO (`TXPIE`) ou o limiar de RXFIFO (`RXPIE`). Além disso, o *software* tem a capacidade de consultar quais interrupções ocorreram por meio do registrador de estado `SPI_SR`. Após o reconhecimento dos eventos, as *flags* podem ser limpas utilizando o registrador `SPI_IFCR`. Esse processo garante que o sistema esteja sempre pronto para detectar e responder a novos eventos de forma eficaz, mantendo a comunicação SPI robusta e eficiente.

spi3_it exti_spi3_wkup	58	51	SPI3	SPI3 global interrupt	0x0000 010C

spi1_it exti_spi1_it	42	35	SPI1	SPI1 global interrupt	0x0000 00CC
spi2_it exti_spi2_it	43	36	SPI2	SPI2 global interrupt	0x0000 00D0

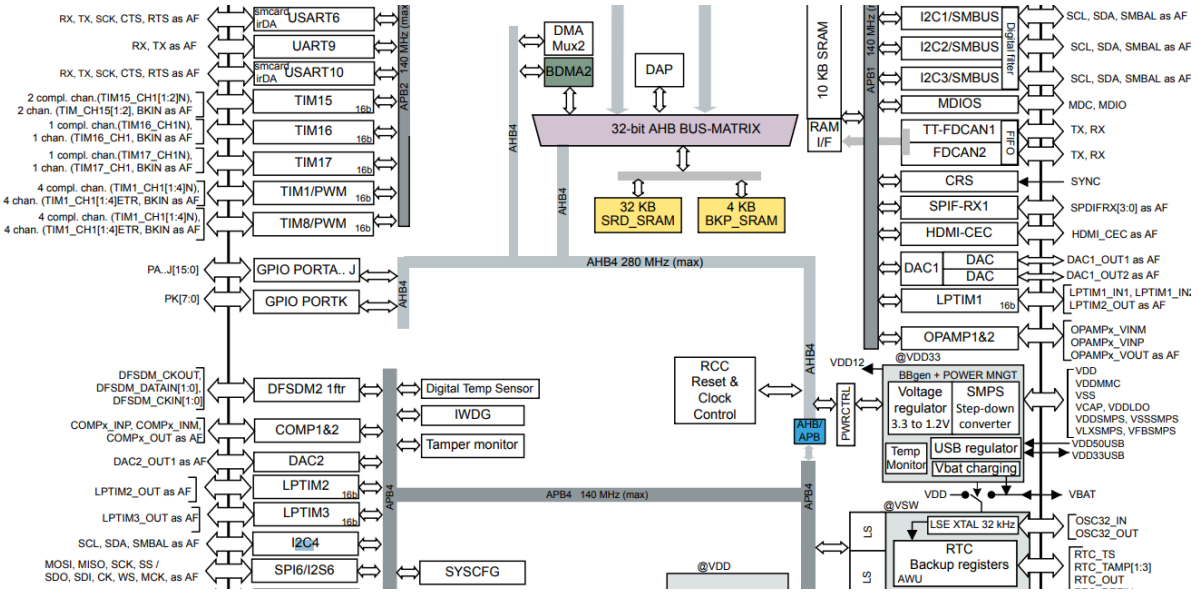
spi4_it exti_spi4_wkup	91	84	SPI4	SPI4 global interrupt	0x0000 0190
spi5_it exti_spi5_wkup	92	85	SPI5	SPI5 global interrupt	0x0000 0194
spi6_it exti_spi6_wkup	93	86	SPI6	SPI6 global interrupt	0x0000 0198

No Manual de Referência são apresentados 2 exemplos de aplicação que envolvem o uso do DMA.

Módulo I2C

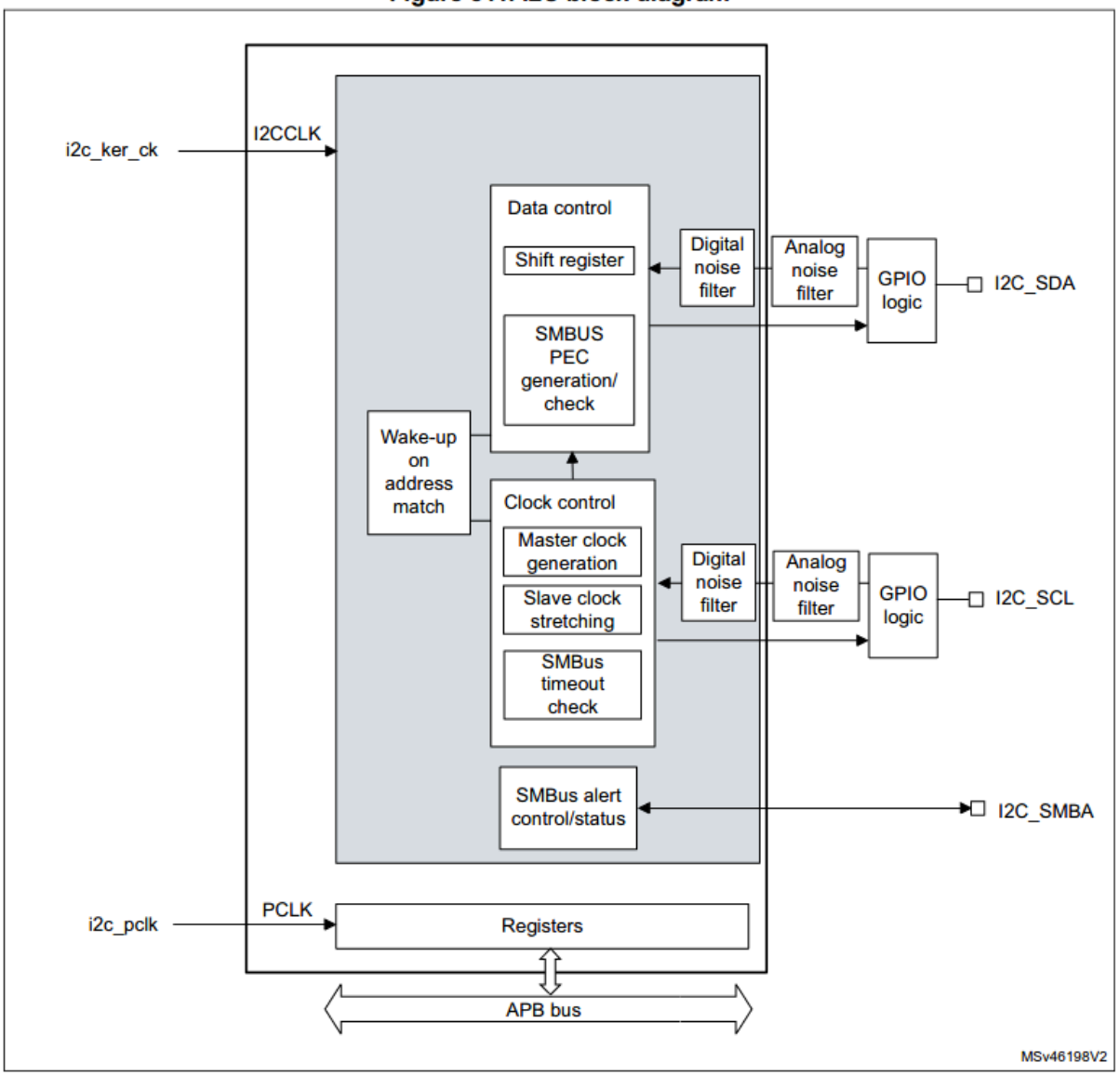
O STM32H7A2 suporta quatro periféricos I2C: I2C1, I2C2, I2C3 e I2C4. Os recursos disponíveis nesses periféricos são compatíveis com a especificação SMBus versão 3.0, que é um protocolo de comunicação usado em dispositivos que requerem monitoramento e controle, como sensores e dispositivos de gerenciamento de energia.

A ativação do *clock gating* para os periféricos I2C envolve uma série de configurações nos registradores RCC (do inglês, *Reset and Clock Control*). O procedimento para habilitar esses periféricos pode ser resumido nas seguintes etapas. Primeiramente, é necessário habilitar o *clock* do periférico I2C (`i2c_pclk`) desejado. Isso é feito configurando o *bit* correspondente no registrador RCC_APB1LENR para I2C1, I2C2 ou I2C3, ou utilizando o *bit* RCC_APB4ENR_I2C4 para o I2C4. Após habilitar o *clock*, é importante garantir que ele permaneça ativo durante o modo de suspensão (CSleep). Para isso, devem ser configurados os *bits* apropriados nos registradores RCC_APB1LLPENR e RCC_APB4LPENR. Outra etapa crucial é a configuração do *clock kernel* (`i2c_ker_ck`). Os periféricos I2C requerem um *clock* rápido o suficiente para atingir a taxa de transmissão desejada. Cada periférico I2C possui um conjunto específico de *bits* nos registradores RCC que controla sua fonte de *clock*. Para selecionar a fonte do *clock kernel* para o I2C1, I2C2 e I2C3, utiliza-se o campo RCC_CDCCIP2R_I2C123SEL, enquanto para o I2C4, a seleção é feita pelos *bits* no campo RCC_SRDCCIPR_I2C4SEL. Esses *bits* permitem escolher entre várias fontes, como RCC, PLLs, HSI e CSI, garantindo que a configuração atenda aos requisitos de desempenho e consumo de energia da aplicação. Por fim, o periférico I2C4 possui um modo autônomo que permite que continue operando mesmo quando a CPU está em modo de baixo consumo, como CStop. Para habilitar esse modo autônomo, deve-se setar o *bit* RCC_SRDAMR_I2C4AMEN em '1'. No modo autônomo, o periférico é capaz de gerar uma requisição de *clock kernel* quando necessário, otimizando assim o consumo de energia.



Para garantir transmissões no protocolo I2C, deve-se ativar o *clock gating* dos pinos GPIO multiplexáveis, configurando-os adequadamente para o modo alternativo. A definição correta da função alternativa nos registradores GPIOx_AFRL e GPIOx_AFRH permite estabelecer a conexão adequada entre os pinos e os periféricos I2C. Na Figura 511 do Manual de Referência, é possível observar que os pinos I2C_SDA e I2C_SCL estão conectados a um circuito de lógica GPIO, que oferece opções para a lógica de saída dos pinos configurados. É importante configurar ambos os pinos como "*open drain*" usando registradores GPIOx_OTYPER e ter resistores de *pull-up* ativados para SDA e SCL, externamente ou por meio da configuração dos registradores GPIOx_PUPDR. Essa configuração é necessária devido à natureza *multi-master* do protocolo I2C, que permite que vários dispositivos compartilhem o mesmo barramento. Com a configuração *open drain*, apenas um dispositivo pode conduzir a linha para baixo de cada vez, enquanto os demais dispositivos permanecem em alta impedância, deixando a linha flutuante. Isso evita conflitos na condução da linha e possíveis curtos-circuitos que poderiam danificar os dispositivos, caso pinos configurados como *push-pull* tentassem conduzir a linha para níveis lógicos diferentes. Dessa forma, a comunicação no barramento I2C se torna mais confiável e segura.

Figure 511. I2C block diagram



As tabelas de funções alternativas dos pinos fornecem uma lista detalhada dos pinos GPIO disponíveis para cada função de periférico. Com base nessas tabelas, é possível selecionar os pinos multiplexáveis para as linhas SDA e SCL em cada periférico I2C. Por exemplo, o par de pinos (PB8, PB9) pode ser configurado para a função alternativa 4 (AF4), que corresponde a I2C1_SCL e I2C1_SDA. Da mesma forma, o par (PB10, PB11) também pode ser configurado para a função alternativa 4 (AF4), correspondente a I2C2_SCL e I2C2_SDA. Além disso, é importante destacar que o par (PB8, PB9) pode ser multiplexado para as funções I2C4_SCL e I2C4_SDA ao definir a função alternativa como 6 (AF6). Essa flexibilidade na configuração

O procedimento básico para garantir uma comunicação I2C confiável e eficiente envolve várias etapas cruciais. O primeiro passo é habilitar o *clock* do periférico I2C desejado, bem como o *clock* do módulo GPIO responsável pelos pinos do periférico I2C. Isso deve ser feito utilizando os registradores RCC apropriados, garantindo que ambos os componentes estejam prontos para operar de forma integrada. Em seguida, recomenda-se realizar um *reset* de *software* no periférico I2C. Isso é importante para assegurar que o dispositivo inicie em um estado conhecido. O *reset* pode ser efetuado definindo o *bit* de *reset* correspondente no registrador `RCC_APB1LRSTR` para os módulos I2C1, I2C2 e I2C3 e `RCC_APB4RSTR` para I2C4. A configuração adequada dos pinos GPIO utilizados na comunicação I2C, como SDA e SCL, é outro passo crítico. Para isso, é necessário definir a função alternativa desses pinos utilizando os registradores `GPIOx_AFRL` ou `GPIOx_AFRH` e configurar os pinos em modo *open-drain* com resistores de *pull-up* ativados.

Após a configuração dos pinos, é necessário ajustar os parâmetros de comunicação I2C utilizando dois registradores de configuração: `I2C_CR1` e `I2C_CR2`. O registrador `I2C_CR1` permite habilitar diversas funcionalidades, como o filtro analógico (`ANFOFF`), o filtro digital para os pinos SCL e SDA (`DNF`), e a configuração do modo DMA (`RXDMAEN`, `TXDMAEN`). Por sua vez, o registrador `I2C_CR2` é responsável por configurar o modo de endereçamento, que pode ser de 7 *bits* ou 10 *bits* (ADD10). O `I2C_CR2` permite definir a quantidade de *bytes* a serem transferidos (`NBYTES`), gerar condições de início (`START`) e fim (`STOP`), determinar o sentido da transferência (`RD_WRN`) e especificar o endereço do escravo (`SADD`). Por padrão, o periférico I2C opera em modo escravo. Muda-se automaticamente de escravo para mestre quando gera uma condição de START e de mestre para escravo se ocorrer uma perda de arbitragem ou a geração de um STOP.

Os registradores `I2C_OAR1` e `I2C_OAR2` são essenciais na configuração dos endereços de dispositivos que operam no modo escravo do barramento I2C. Os endereços definidos nesses registradores são utilizados para identificar o periférico escravo durante a comunicação. Quando um mestre I2C inicia uma transmissão, ele envia o endereço do escravo desejado, configurado no campo `I2C_CR2_SADD`. Se o *bit* `I2C_OAR1_OA1EN` estiver definido como "1", o escravo compara o endereço recebido com o endereço configurado no campo `I2C_OAR1_OA1`, adotando o modo de endereçamento (7 ou 10 *bits*) especificado no *bit* `I2C_OAR1_OA1MODE`. Se houver correspondência, o escravo responde, estabelecendo a comunicação. Caso contrário, se endereços adicionais estiverem configurados (`I2C_OAR1_OA2EN` = 1), o periférico escravo poderá responder ao endereço definido em `I2C_OAR2_OA2` e a até sete variantes geradas pela máscara definida no campo

`I2C_OAR2_OA2MSK`, possibilitando a comunicação. A correta configuração dos endereços nos registradores `I2C_OAR1` e `I2C_OAR2` é fundamental para o funcionamento adequado da comunicação I2C. Um endereço configurado incorretamente impede que o escravo responda ao mestre, impossibilitando a troca de dados.

O registrador `I2C_TIMINGR` é usado para configurar a temporização do I2C, com campos que especificam tempos de *setup* (SCLDEL) e *hold* (SDADEL) de dados, largura de nível alto de SCL (SCLH) largura de nível baixo de SCL (SCLL), e um prescaler (`PRESC`). Esses tempos são ilustrados nos diagramas de tempo a seguir.

Figure 513. Setup and hold timings

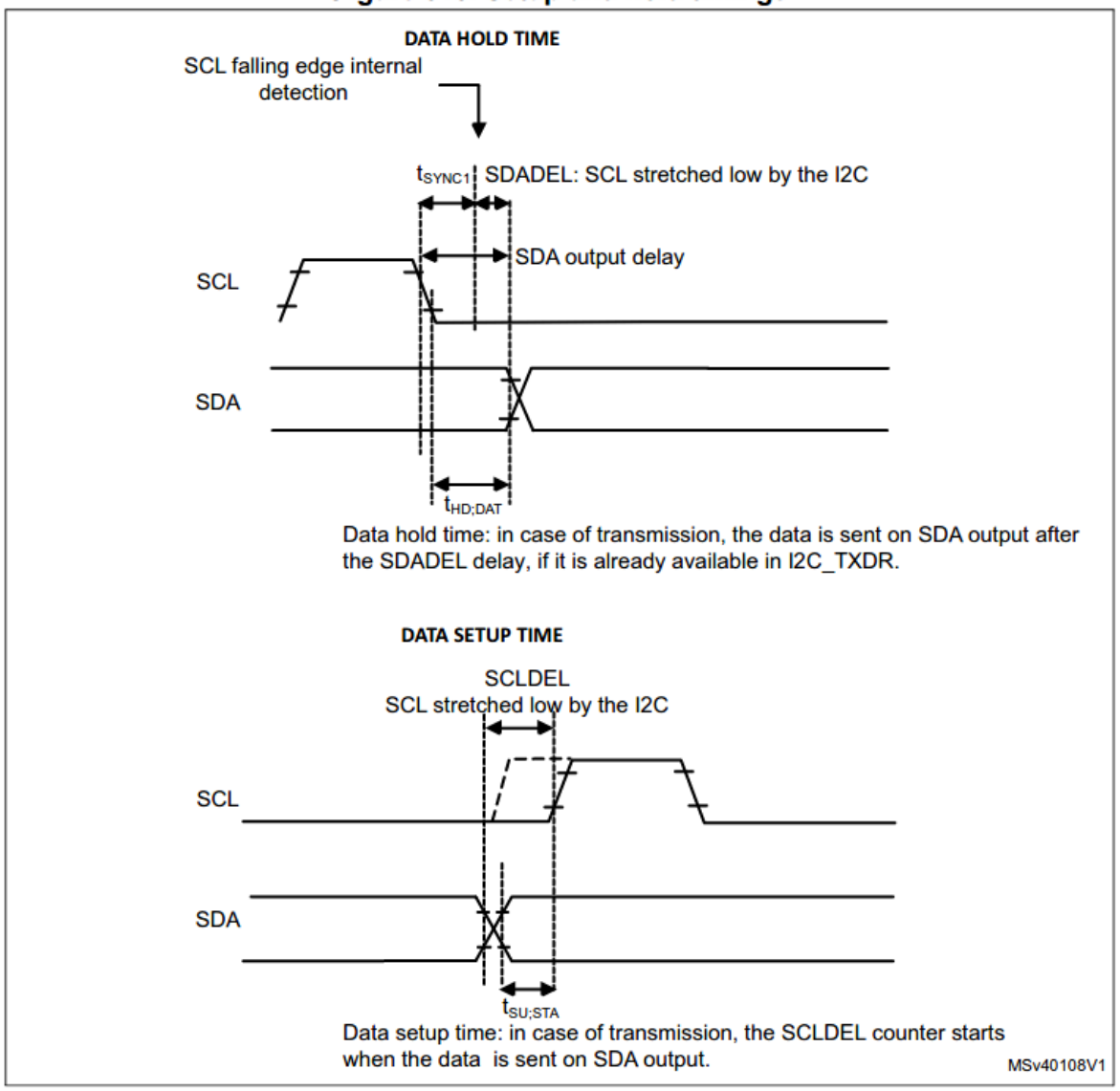
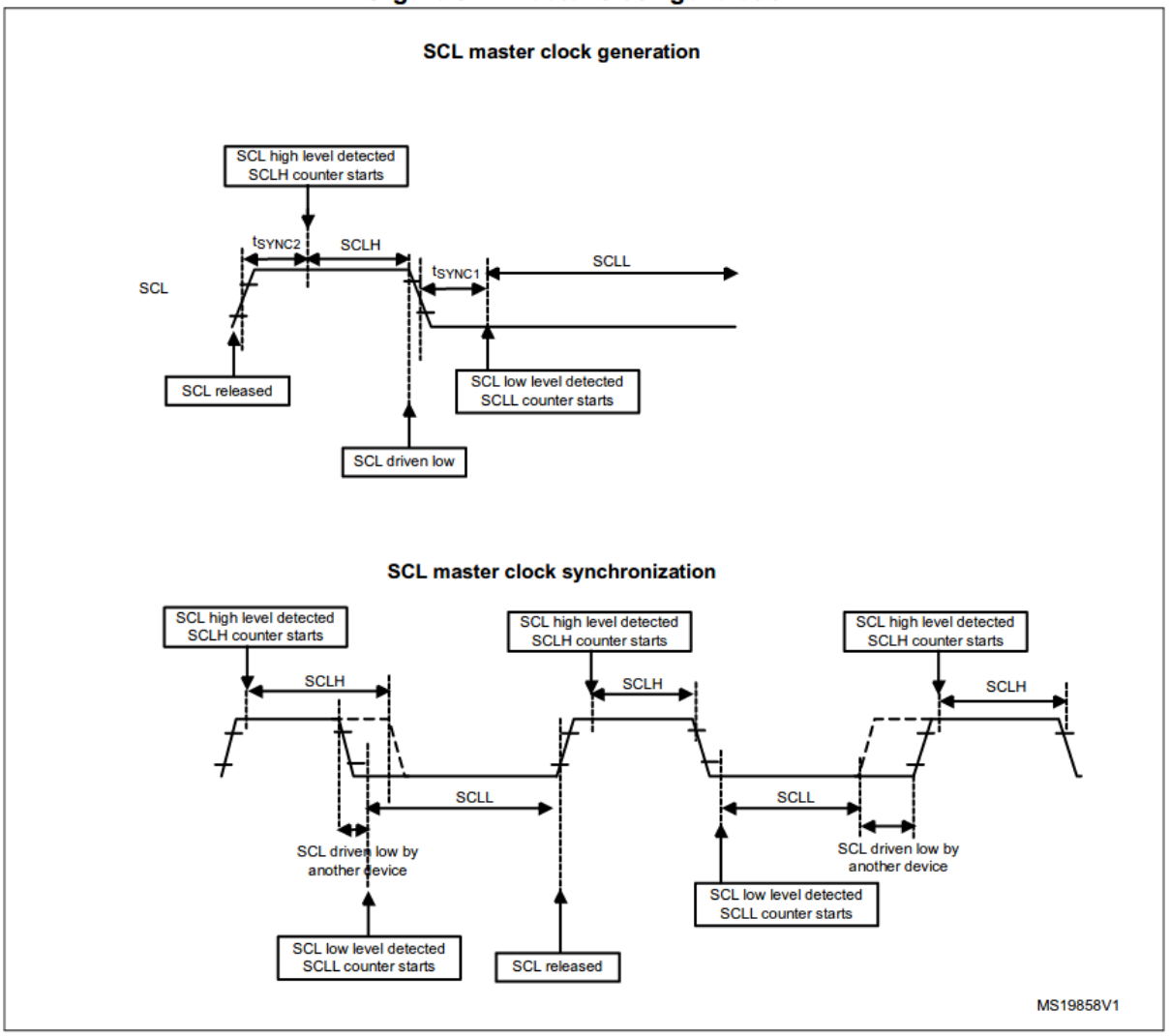


Figure 524. Master clock generation



A partir da frequência do *clock kernel*, i2c_ker_ck e dos parâmetros configurados no registrador `I2C_TIMINGR`, podemos derivar as seguintes métricas de tempo:

$$t_{PRESC} = \frac{PRESC+1}{i2c\_ker\_ck} = (PRESC+1) \times t_{I2CCLK}$$

$$t_{SCLDEL} = (SCLDEL+1)\, x\, t_{PRESC}$$

$$t_{SDADEL} = SDADEL \times t_{PRESC}$$

$$t_{SCLH} = (SCLH+1) \times t_{PRESC}$$

$$t_{SCLL} = (SCLL+1) \times t_{PRESC}$$

Essas medidas devem satisfazer as restrições temporais estabelecidas pelo protocolo I2C e SMBus.

Table 376. I²C-SMBus specification clock timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency	-	100	-	400	-	1000	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	0.26	-	4.7	-	
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	
t_{LOW}	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	
t_{HIGH}	Period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

A Seção 52.4.10 do Manual de Referência apresenta alguns exemplos de configuração dos campos do registrador `I2C_TIMINGR` para difentes frequências de *clock kernel* e demandas

Por fim, para que o periférico I2C comece a operar, é necessário habilitá-lo definindo o *bit* `I2C_CR1_PE` no registrador `I2C_CR1`. Com essa etapa concluída, o periférico estará pronto para iniciar a comunicação I2C. Para informações detalhadas sobre a configuração do I2C, consulte o Manual de Referência. As seções relevantes incluem: configuração do modo escravo e processamentos no modo escravo (Seção 52.4.8); e configuração do modo mestre e processamentos no modo mestre (Seção 52.4.9).

Figura 515 do Manual de Referência ilustra a dinâmica de recepção de um dado. Os dados de entrada SDA são armazenados em um registrador de deslocamento e transferidos para o registrador `I2C_RXDR`, caso este esteja vazio (`I2C_ISR_RXNE` = 0). Se `I2C_ISR_RXNE` = 1, significando que o *byte* de dados recebido anteriormente ainda não foi lido, a linha SCL é estendida até que o `I2C_RXDR` seja lido.

Figure 515. Data reception

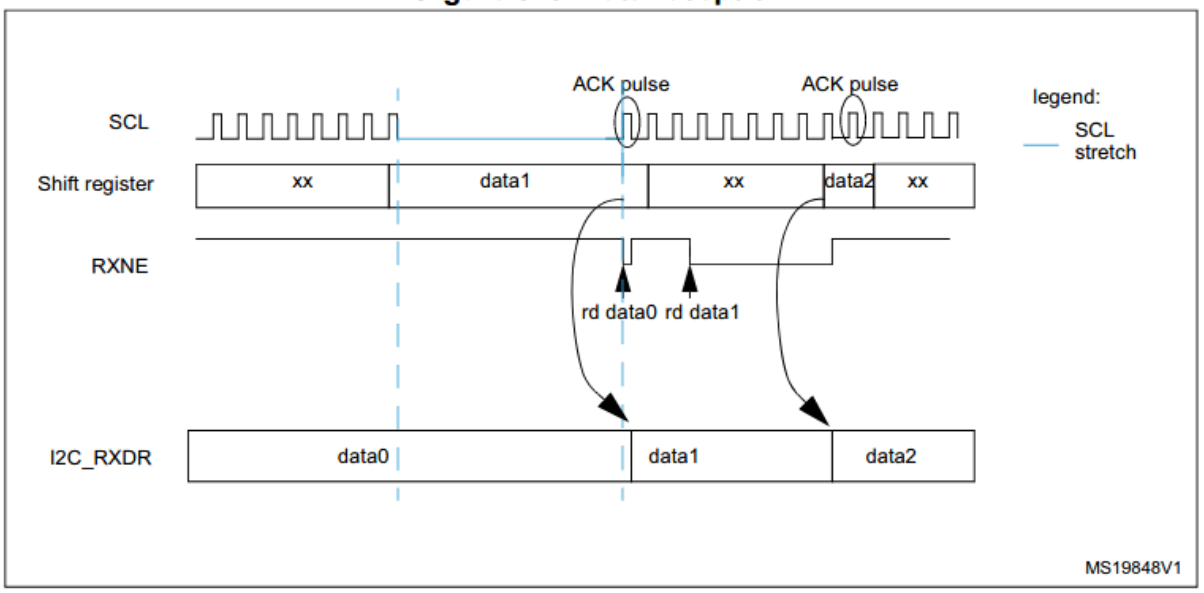
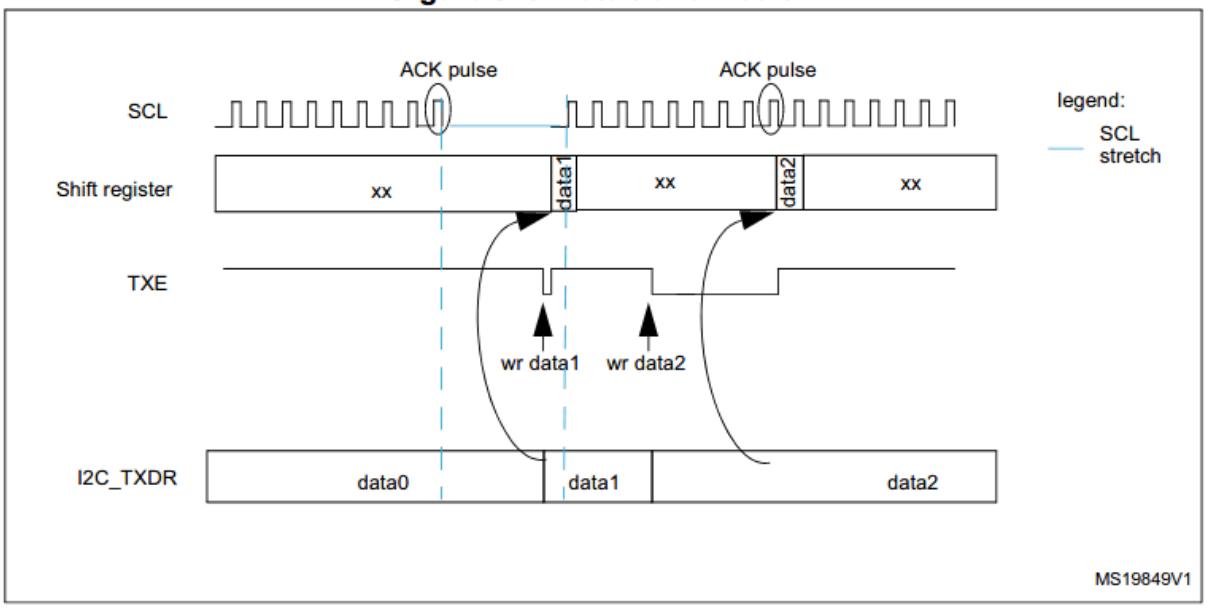


Figura 516 do Manual de Referência ilustra a transmissão dos dados armazenados no registrador `I2C_TXDR`. Se o registrador I2C_TXDR não estiver vazio (`I2C_ISR_TXE` = 0), seu conteúdo é copiado para o registrador de deslocamento após o nono pulso SCL (o pulso de reconhecimento). Em seguida, o conteúdo do registrador de deslocamento é deslocado para fora na linha SDA. Se `I2C_ISR_TXE` = 1, significando que nenhum dado foi escrito ainda no `I2C_TXDR`, a linha SCL é estendida até que o I2C_TXDR seja escrito.

Figure 516. Data transmission



O fluxo de dados entre o periférico I2C e a memória pode ser otimizado por meio do uso do DMA. Sem o DMA, a CPU é responsável por gerenciar cada *byte* transferido, o que consome tempo e pode comprometer o desempenho, especialmente em aplicações que lidam com grandes volumes de dados. Com o DMA, a transferência de dados é delegada ao controlador DMA e ao multiplexador DMAMUX, permitindo que ambos operem sem intervenção da

CPU, liberando-a para outras tarefas. A configuração do controlador DMA e do DMAMUX é semelhante àquela descrita para a transferência de dados entre os periféricos SPI e a memória via DMA. Basta consultar a tabela de mapeamento do DMAMUX e substituir os códigos (*DMA request MUX input*) correspondentes aos periféricos SPI pelos respectivos códigos dos periféricos I2C e habilitar transferências via DMA sentando os *bits* `I2C_CR1_RXDMAEN` e `I2C_CR1_TXDMAEN` em '1'. Os I2C1, I2C2 e I2C3 são mapeados no DMAMUX1, enquanto o I2C4 é mapeado no DMAMUX2.

13	I2C4_RX	29	Reserved
14	I2C4_TX	30	Reserved

31	TIM4_CH3	73	I2C3_RX	115	Reserved
32	TIM4_UP	74	I2C3_TX	116	UART9_RX
33	I2C1_RX	75	DCMI_PSSI	117	UART9_TX
34	I2C1_TX	76	CRYP_IN	118	USART10_RX
35	I2C2_RX	77	CRYP_OUT	119	USART10_TX
36	I2C2_TX	78	HASH_IN	120	Reserved

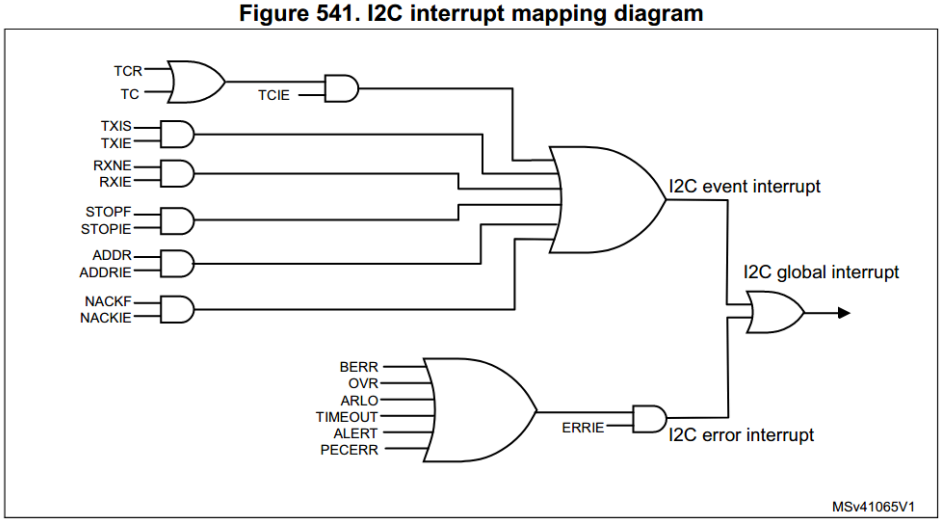
Os periféricos I2C geram diversos eventos de interrupção que sinalizam diferentes estados e ocorrências durante a comunicação. Esses eventos podem ser habilitados e configurados para gerar interrupções (IRQs) no processador, permitindo que o *software* reaja a situações específicas. A Tabela 386 do Manual de Referência lista os eventos de interrupção disponíveis para os periféricos I2C. A segunda coluna da tabela apresenta o *bit* que indica o estado do evento correspondente no registrador `I2C_ISR`. A terceira coluna descreve as ações necessárias para limpar cada *bit* de estado, além do uso do registrador de limpeza `I2C_ICR`. Por fim, a quarta coluna indica o *bit* de habilitação de interrupção no registrador `I2C_CR1`.

Table 386. I2C interrupt requests

Interrupt event	Event flag	Event flag/interrupt clearing method	Interrupt enable control bit	Interrupt/wake-up activated		
				i2c_event_it	i2c_error_it	i2c_wkup
Receive buffer not empty	RXNE	Read I2C_RXDR register	RXIE	Yes	No	No
Transmit buffer interrupt status	TXIS	Write I2C_TXDR register	TXIE			
Stop detection interrupt flag	STOPF	Write STOPCF=1	STOPIE			
Transfer Complete Reload	TCR	Write I2C_CR2 with NBYTES[7:0] ≠ 0	TCIE			
Transfer complete	TC	Write START=1 or STOP=1				
Address matched	ADDR	Write ADDRCF=1	ADDRIE			Yes(1)
NACK reception	NACKF	Write NACKCF=1	NACKIE			No
Bus error	BERR	Write BERRCF=1	ERRIE	N0	Yes	
Arbitration loss	ARLO	Write ARLOCF=1				
Overrun/underrun	OVR	Write OVRCF=1				
PEC error	PECERR	Write PECERRCF=1				
Timeout/t_{LOW} error	TIMEOUT	Write TIMEOUTCF=1				
SMBus Alert	ALERT	Write ALERTCF=1				

Quando uma interrupção é gerada pelo periférico I2C e a interrupção correspondente está habilitada, é gerada uma solicitação de interrupção por evento (em inglês, *I2C event interrupt*) ou uma solicitação de interrupção por erro (em inglês, *I2C error interrupt*) ao controlador NVIC (do inglês, *Nested Vector Interrupt Controller*).

Figure 541. I2C interrupt mapping diagram



Segue-se o mapeamento das 2 IRQs de cada periférico I2C nos números de vetores de interrupção.

Signal	Priority	NVIC position	Acronym	Description	Address offset
i2c2_ev_it exti_i2c2_ev_wkup	40	33	I2C2_EV	I2C2 event interrupt	0x0000 00C4
i2c2_err_it	41	34	I2C2_ER	I2C2 error interrupt	0x0000 00C8
i2c1_ev_it exti_i2c1_ev_wkup	38	31	I2C1_EV	I2C1 event interrupt	0x0000 00BC
i2c1_err_it	39	32	I2C1_ER	I2C1 error interrupt	0x0000 00C0
i2c3_ev_it exti_i2c3_ev_wkup	79	72	I2C3_EV	I2C3 event interrupt	0x0000 0160
i2c3_err_it	80	73	I2C3_ER	I2C3 error interrupt	0x0000 0164
i2c4_ev_it exti_i2c4_ev_it	102	95	I2C4_EV	I2C4 event interrupt	0x0000 01BC
i2c4_err_it	103	96	I2C4_ER	I2C4 error interrupt	0x0000 01C0