

Usando uma LLM preferida, aprenda um pouco mais sobre os esquemas de codificação.

### 1) Quando é recomendado utilizar NRZ e NRZ-M?

Boa pergunta! As codificações NRZ (Non-Return-to-Zero) e NRZ-M (Non-Return-to-Zero Mark) são esquemas de **codificação de linha** usados para representar bits digitais em sinais elétricos. Cada uma tem usos mais recomendados dependendo do tipo de canal, sincronização e simplicidade desejada.

#### ✓ NRZ (Non-Return-to-Zero)

##### 🔧 Como funciona:

- Bit **1** : nível alto constante.
- Bit **0** : nível baixo constante.
- O sinal **não retorna ao zero entre bits**, daí o nome.

##### 🟢 Recomendado quando:

- O canal tem **baixo ruído** e **boa sincronização de clock**.
- A largura de banda precisa ser **conservada** (baixa frequência).
- O receptor já tem um **clock síncrono** confiável.
- Transmissões de **curta distância**, como entre chips ou em barramentos digitais.

##### 🔴 Problemas:

- Longas sequências de **0** s ou **1** s causam **ausência de transições**, dificultando a **sincronização de clock**.
- Não tem boa detecção de erros embutida.

## ✓ NRZ-M (Non-Return-to-Zero Mark)

### 🔧 Como funciona:

- Uma **transição de nível** (inverte) representa um bit **1**.
- **Nenhuma transição** representa um bit **0**.
- Ou seja, o bit **1** é um "mark" (marca = mudança).

### 🟢 Recomendado quando:

- A **sincronização de clock** é importante.
- Há necessidade de **detecção de mudanças de estado** (e não valor exato).
- Em canais onde **inversões de polaridade** podem ocorrer — o padrão ainda pode ser decodificado corretamente, pois depende de **transições** e não níveis fixos.

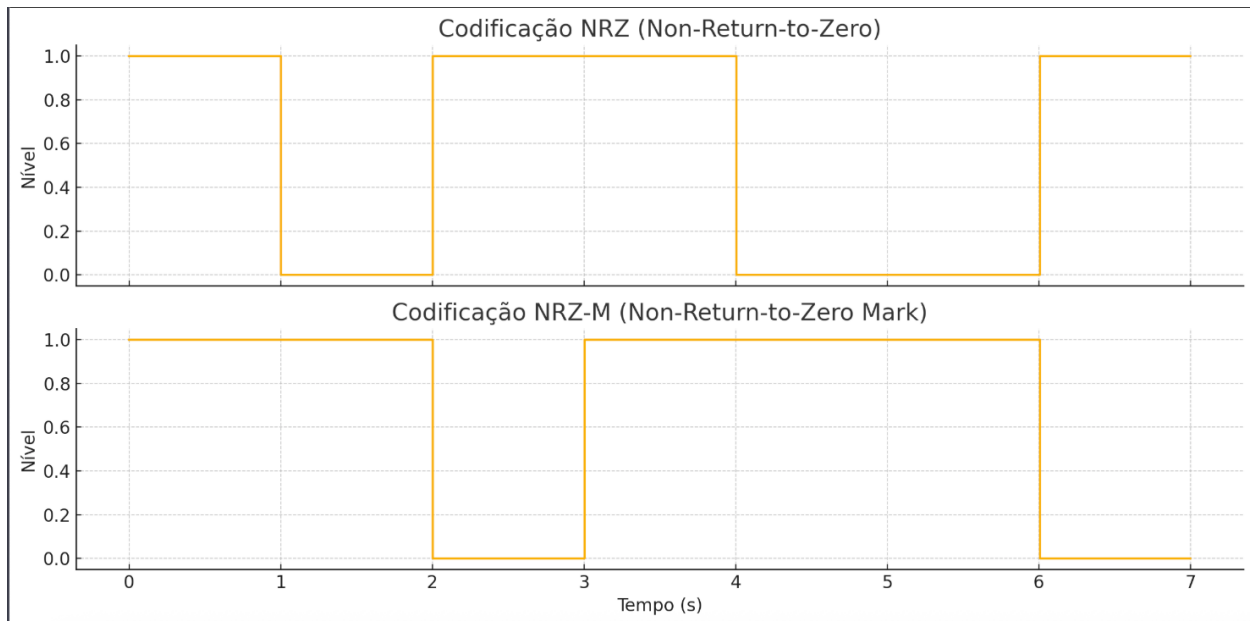
### 🔴 Problemas:

- Pode ser um pouco mais difícil de implementar que o NRZ.
- Em certas sequências com muitos **0**s, ainda pode haver **poucas transições** (apesar de ser melhor que o NRZ puro).

### 🧠 Resumo comparativo:

Característica	NRZ	NRZ-M
Sincronismo	Fraco (sem transições)	Melhor (depende de transições de 1)
Detecção de erro	Fraca	Média
Eficiência espectral	Alta (baixa banda)	Boa
Facilidade	Simples	Um pouco mais complexa
Aplicações típicas	SPI, EEPROM, UART	HDLC, SDLC, protocolos antigos

## 2) Desenhe o funcionamento de cada um para uma sequencia de bits



- **NRZ:** representa o bit **1** com nível alto e o bit **0** com nível baixo, mantendo o mesmo valor durante todo o tempo do bit.
- **NRZ-M:** faz uma transição de nível sempre que o bit é **1** e mantém o nível quando o bit é **0**.

Com isso, dá pra ver como **NRZ-M** ajuda na sincronização, porque sempre que tem um **1**, há uma transição no sinal — mesmo que os níveis absolutos não importem tanto.

### 3) Quando que é recomendável utilizar códigos sensíveis a fase e quando utilizar sensíveis ao sinal?

#### ⚡ Códigos sensíveis ao nível do sinal

➤ Ex: NRZ, RZ, Unipolar, Polar, Bipolar (AMI)

##### ● Recomendado quando:

- O canal é estável e com **baixo ruído**.
- Existe um **clock compartilhado** ou uma boa forma de sincronização externa.
- Há **limitação de largura de banda** (NRZ é mais eficiente).
- A complexidade do sistema deve ser **mínima**.

##### ✓ Vantagens:

- Mais **simples** de implementar.
- Requer **menos largura de banda**.
- Boa eficiência espectral (ex: NRZ).

##### ✗ Desvantagens:

- Podem apresentar **problemas de sincronismo** quando há longas sequências de bits iguais.
- **Menos tolerantes a ruído** do que os sensíveis à fase.

#### 🌐 Códigos sensíveis à fase

➤ Ex: Manchester, Manchester diferencial, PSK (fase em modulação)

##### ● Recomendado quando:

- O canal está sujeito a **inversão de polaridade** ou ruído (ex: comunicação serial, redes físicas instáveis).
- O receptor **precisa extrair o clock do sinal recebido** (não há sincronismo externo).
- A confiabilidade é mais importante que a largura de banda.
- Em **comunicações modernas**, onde o canal é ruidoso ou onde o sistema precisa de autossincronização (ex: RFID, Ethernet antigo, NFC).

##### ✓ Vantagens:

- Excelente para **recuperar clock**.
- Mais **resistentes a erros** causados por inversão de sinal.
- **Transições frequentes** facilitam sincronismo.

##### ✗ Desvantagens:

- Ocupam mais **largura de banda** (ex: Manchester dobra a taxa de sinal).
- São mais **complexos de implementar**.

## Resumo prático:

Situação	Tipo de código mais adequado
Canal limpo, curtas distâncias	Sensível ao sinal (NRZ, RZ)
Canal sujeito a ruído	Sensível à fase (Manchester, PSK)
Precisa extrair o clock	Sensível à fase
Baixa largura de banda disponível	Sensível ao sinal
Comunicação moderna confiável (sem fio, serial)	Sensível à fase

### 4) Como que o ruído afeta os diferentes esquemas de codificação?

#### Como o ruído afeta os esquemas de codificação?


##### 1. Códigos sensíveis ao nível do sinal (ex: NRZ, RZ, Unipolar, Bipolar)

- O ruído pode **alterar o nível do sinal** (de 0 para 1 ou vice-versa), causando **erro direto na leitura do bit**.
- Como essas codificações mantêm o sinal constante durante todo o tempo do bit, **qualquer interferência** nesse intervalo pode gerar **erros de detecção**.
- Exemplo: se uma interferência de alta amplitude ocorre durante um bit **0**, ele pode ser interpretado como **1**.

 **Mais vulneráveis a ruído** — especialmente em canais com ruído branco ou impulsivo.

##### 2. Códigos sensíveis à transição ou fase (ex: Manchester, Manchester diferencial)

- O bit é identificado com base em **mudanças no sinal**, não no nível absoluto.
- Como as transições ocorrem em momentos previsíveis (geralmente no meio do bit), o sistema é capaz de manter a **sincronização do clock** mesmo com ruído.
- Para causar erro, o ruído teria que ocorrer **exatamente no momento da transição**, o que é menos provável.

 **Mais robustos contra ruído** e melhores para canais com interferência, pois não dependem da leitura de nível fixo, mas da presença ou ausência de transição.