

# Laboratório de Conectividade – PTC 3260

---

## Conexão via Enlace de Rádio com NRF24L01+

---



Escola Politécnica da Universidade de São Paulo  
Departamentos da Engenharia Elétrica  
PTC Telecomunicações e Controle

Responsável: Prof. Cristiano Panazio

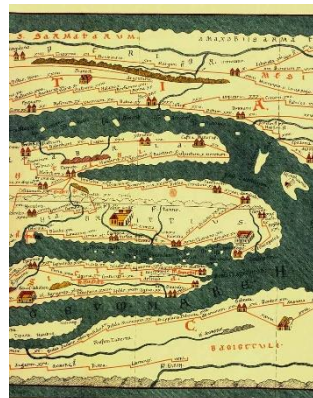
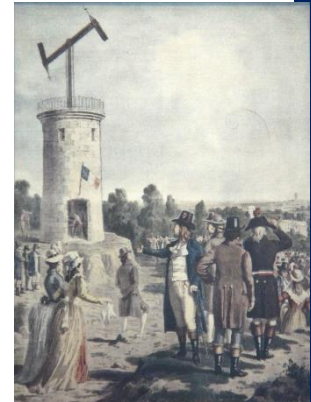


Março de 2024



# 1. Breve Histórico

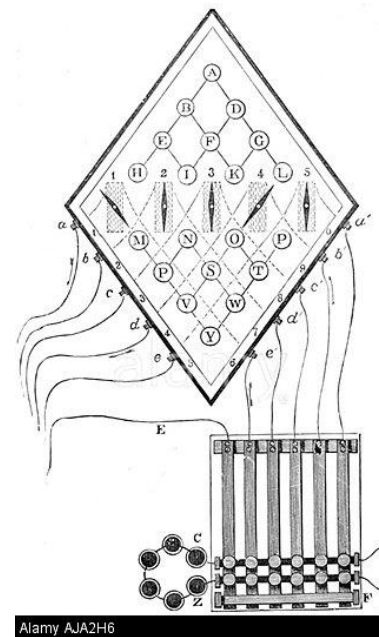
- Como fazer para levar informação por longas distâncias e/ou rapidamente?
  - Meios não elétricos:
    - Visual: Piras em colinas/montanhas para avisar a chegada de inimigos, e posteriormente para o telégrafo (semáforo) óptico no século XVIII (Claude Chappe e Abraham Niclas Edelcrantz)
    - Correio: origem no sistema *angariae* persa e *cursus publicus* dos romanos





# 1. Breve Histórico

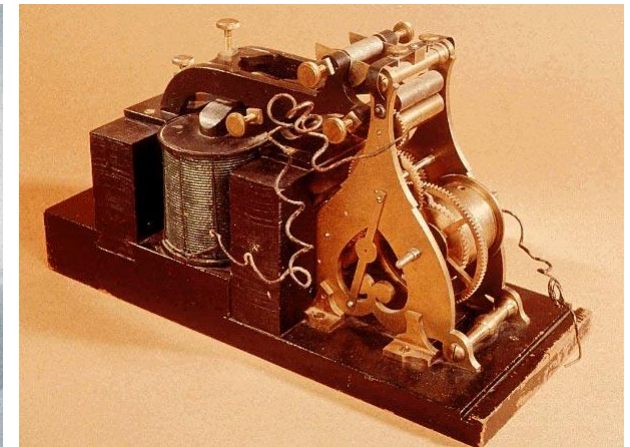
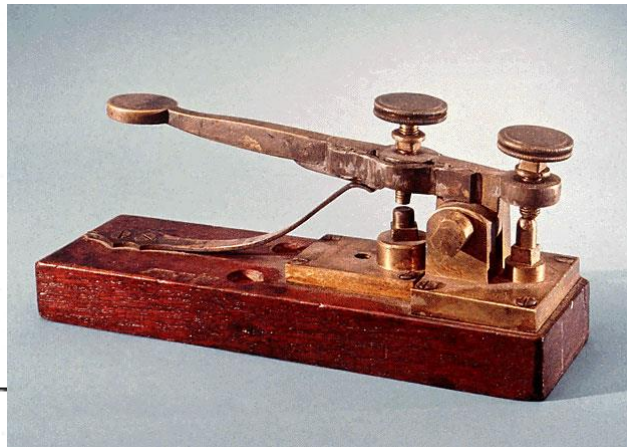
- Eletricidade: com a descoberta da indução eletromagnética por Henry Faraday (1831), temos o telégrafo elétrico de Wheatstone & Cooke (1837):



Alamy AJA2H6

e de Morse (1844):

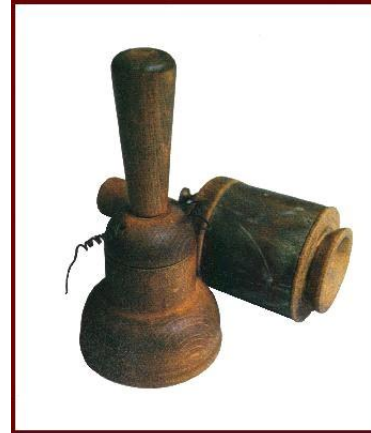
A . —	N — .	& . ...
B — ...	O ..	1 . — — .
C .. .	P .....	2 .. — ..
D — ..	Q .. — .	3 ... — .
E .	R .. .	4 .... —
F . — .	S ...	5 — — —
G — — .	T —	6 .. . . .
H ....	U .. —	7 — — ..
I ..	V ... —	8 — ....
J — . — .	W . — —	9 — .. —
K — . —	X . — ..	0 — — — —
L —	Y .. ..	
M — —	Z ....	



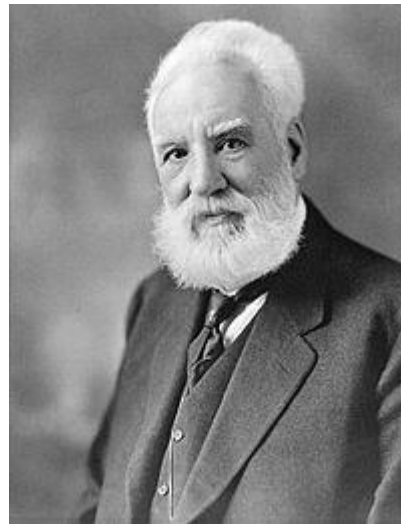
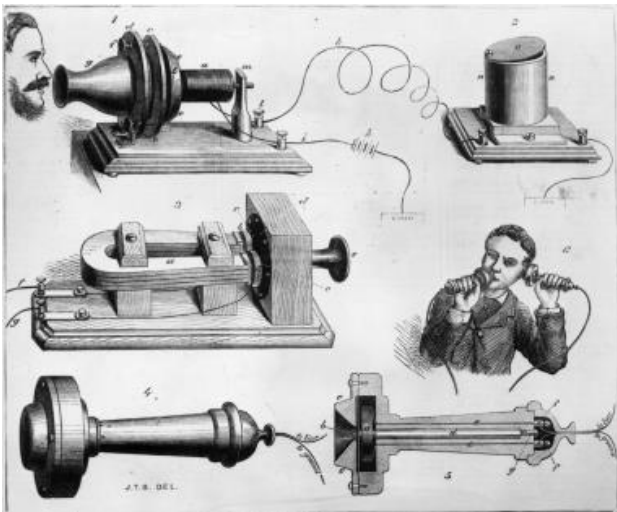


# 1. Breve Histórico

- A comunicação por voz, i.e., o telefone, teve início com várias pessoas, dentre elas Antonio Meucci (1854)



e posteriormente com Alexander G. Bell e Elisha Gray (1876):







# 1. Breve Histórico

- A transmissão de sinais sem fio: depois dos estudos de James Maxwell, Heinrich Hertz e Oliver Lodge, tivemos o telégrafo sem fio de Guglielmo Marconi





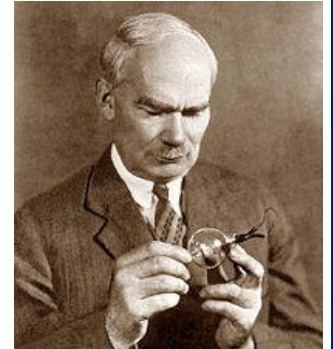
# 1. Breve Histórico

- A evolução continuou passando por descobertas e invenções de:

Joseph Fourier (1822)



Lee de Forest (1907)



Edwin Armstrong (1918 e 1933)



Harry Nyquist (1928)



até chegarmos aos princípios da comunicação digital moderna com...



# 1. Princípio das comunicações digitais

"O problema fundamental da comunicação é reproduzir em um dado ponto, exata ou aproximadamente, uma mensagem produzida em outro ponto."

Claude Elwood Shannon (1916-2001)  
– o pai da Teoria da Informação



- O principal problema é a existência de **ruído**.
- Existem diversos mecanismos para lidar com tal problema.
- Todavia, não há “almoço gratuito” e paga-se com redução da taxa de comunicação e/ou energia gasta pelo sistema



## 2. Alcance do sinal: efeitos da potência e taxa de transmissão

- Objetivo: transmitir com maior alcance possível, mantendo-se uma boa qualidade do sinal recebido (i.e., nenhum erro na recepção)
- Sabe-se que o receptor usa um integrador (com janela de duração do bit) para estimar os bits na presença de ruído
  - Isto permite tomar a decisão não apenas com um único valor do sinal recebido a cada bit, mas com todo o sinal que forma cada um dos bits transmitidos

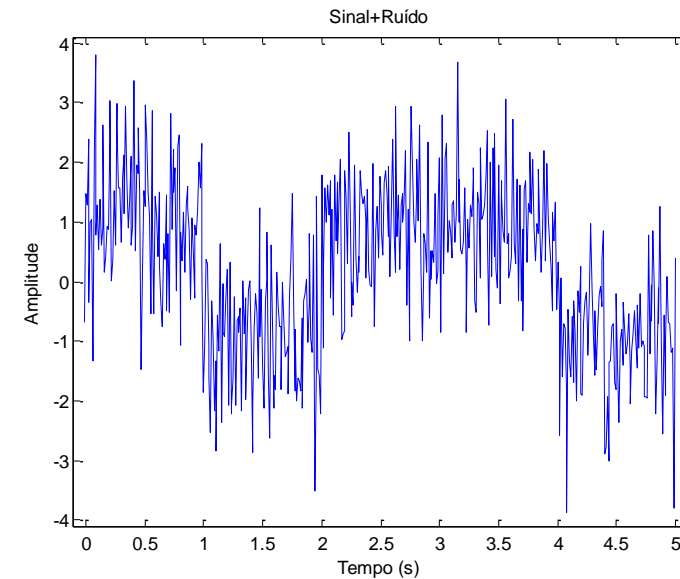
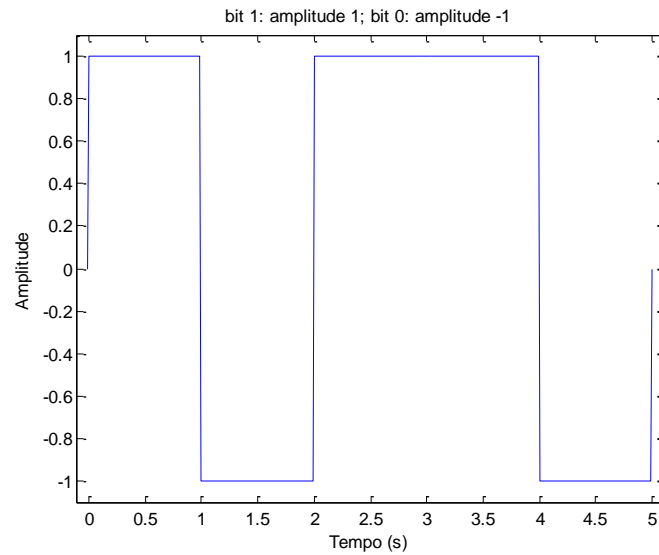
Algumas soluções :  
(não exaustivo e não excludentes)



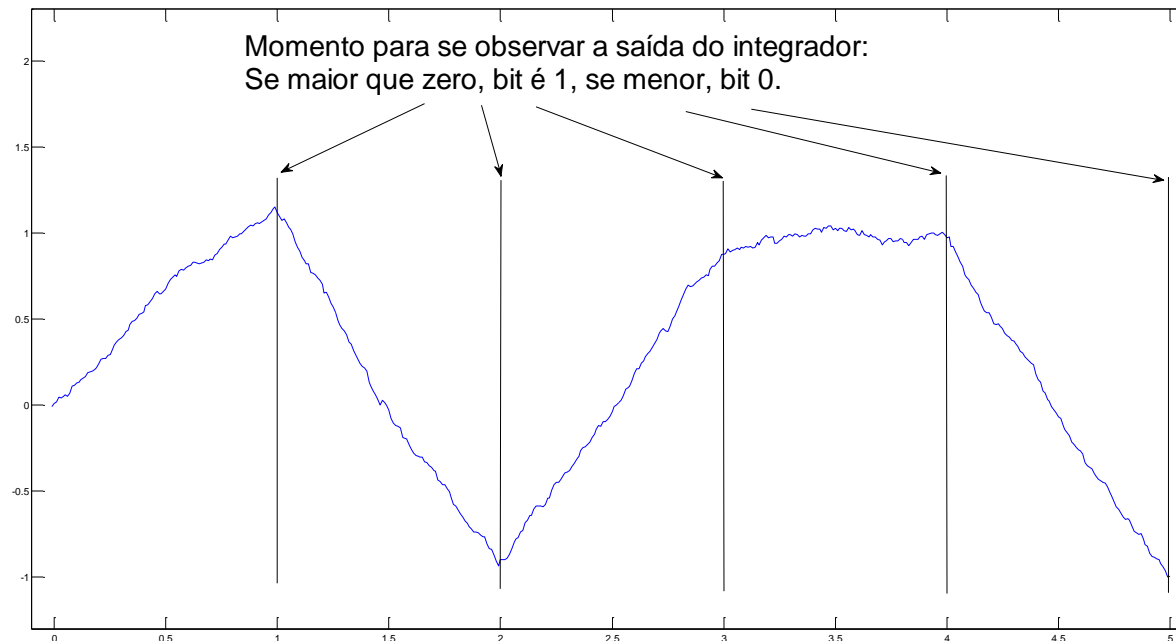


## 2. Alcance do sinal: efeitos da potência e taxa de transmissão

- Ex:



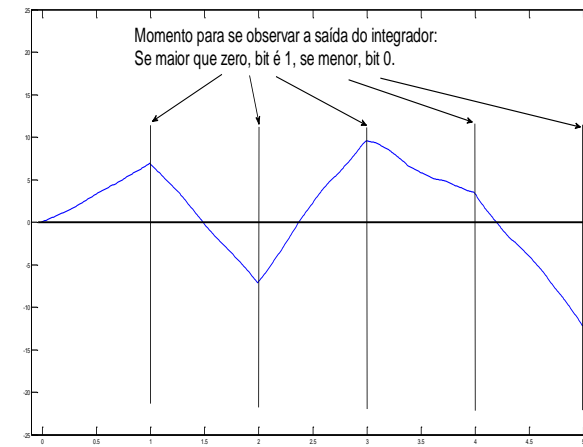
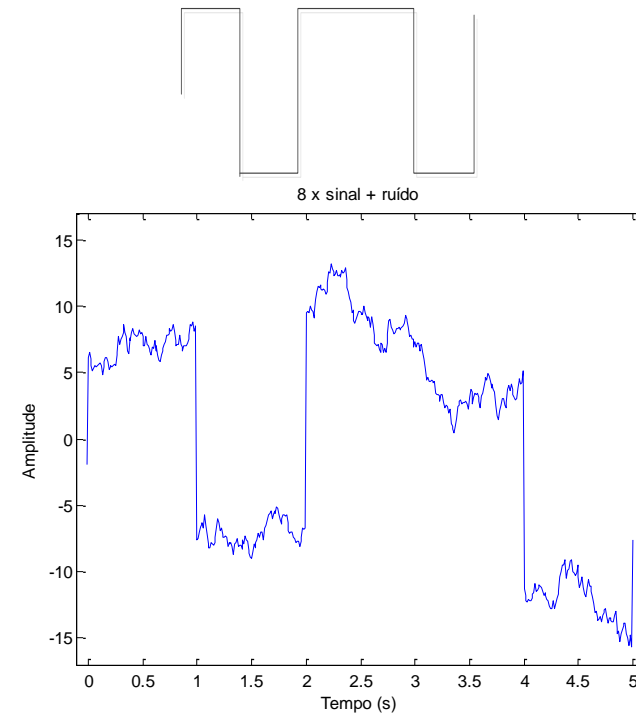
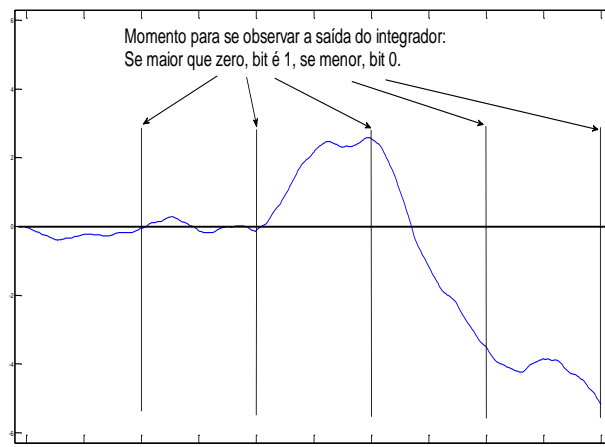
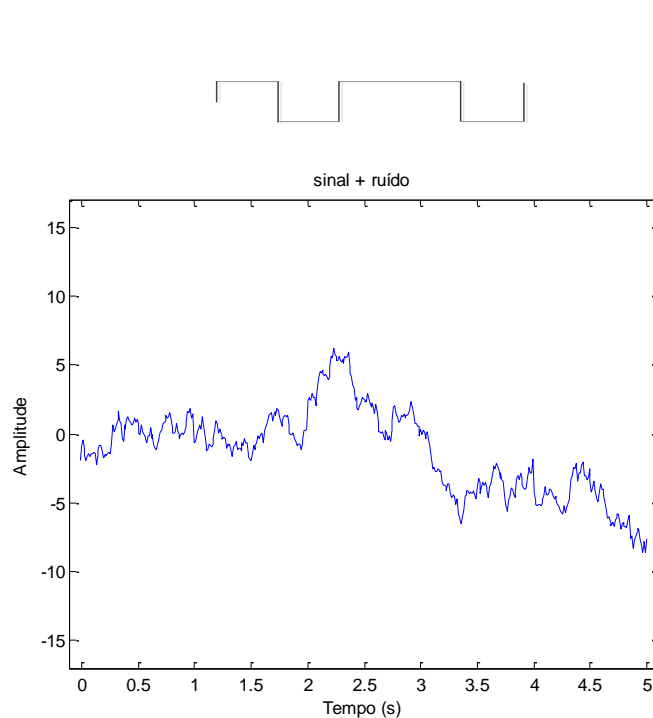
- Integrando:





## 2. Alcance do sinal: efeitos da potência e taxa de transmissão

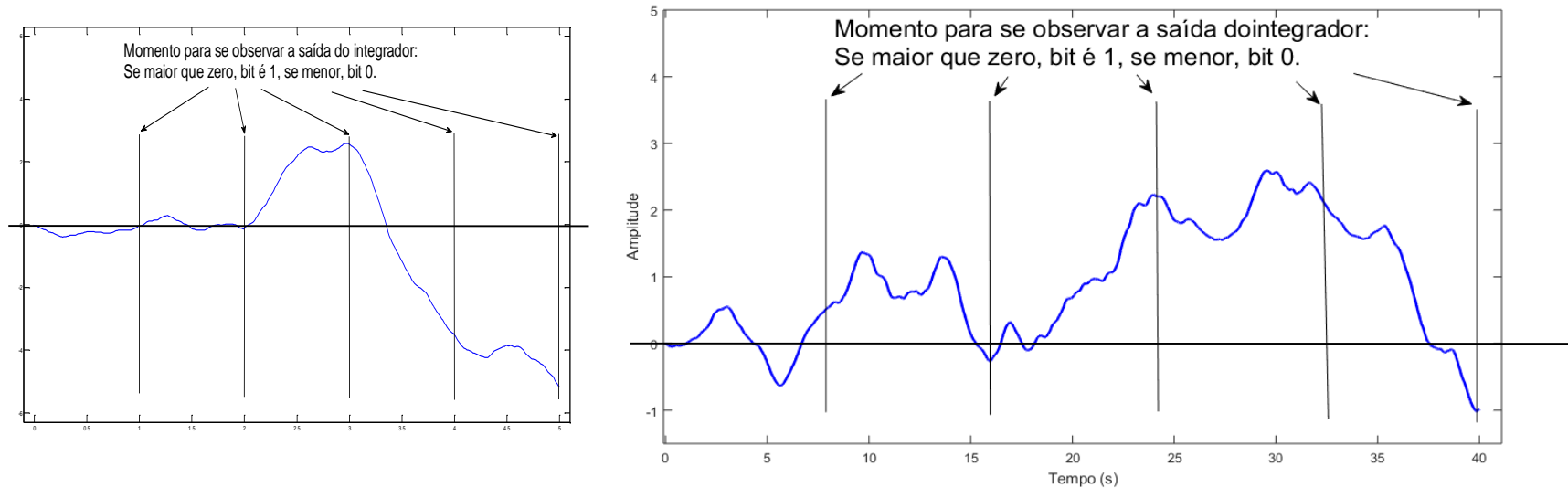
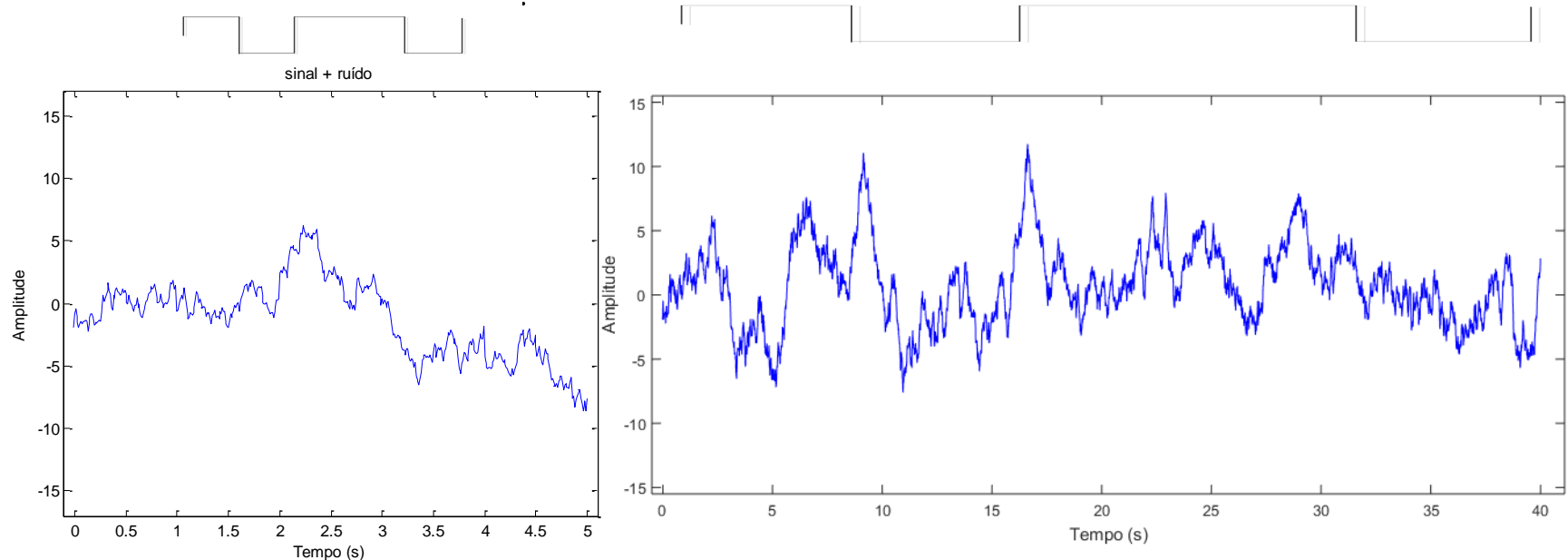
- Aumentar a potência (amplitude) de transmissão: o sinal chega com mais “força” (energia) no receptor, aumentando a robustez ao ruído, mas, gasta-se mais energia, diminuindo, por exemplo, a autonomia do sistema





## 2. Alcance do sinal: efeitos da potência e taxa de transmissão

- **Diminuir a taxa de transmissão:** coloca-se mais energia nos bits transmitidos, aumentando a robustez ao ruído, mas a taxa de transmissão é reduzida e se está mais sujeito a colisões (sobreposição de transmissões no receptor)





### 3. Lidando com erros

- Mesmo operando com potência e taxas adequadas, o sinal recebido não está completamente livre de erros
- Um erro pode ser fatal (*e.g.*, errar o bit mais significativo numa transação bancária levando a milhões de prejuízo)

#### Ideia: detectar com alto grau de certeza a existência de tais erros

Usar alguma redundância na transmissão na forma de um código (bits gerados a partir da informação transmitida) e checar na recepção se tais bits estão de fato condizentes com a informação recebida

- Não há distinção entre erros, sejam aqueles que ocorrem na informação ou na própria redundância
- Em caso de erro, i) pedir retransmissão, ou ii) corrigir o erro se o código permitir
- A chance de um erro passar despercebido deve ser muito pequena



### 3. Lidando com erros

#### – Exemplo: Cadastro de Pessoa Física (CPF)

Os dois últimos dígitos são ditos dígitos verificadores, ou de paridade (i.e., redundância)

Pseudo código:

Verificação: temp1 e temp2 são, respectivamente, o primeiro e segundo dígitos verificadores)

```
for (int k=0, k<=8, k++) {  
    temp1+=(k+1)*cpf[k];  
}
```

```
temp1=mod(temp1,11);
```

```
temp1=mod(temp1,10);
```

```
for (int k=1, k<=9, k++) {  
    temp2+=(k)*cpf[k];  
}
```

```
temp2=mod(temp2,11);
```

```
Temp2=mod(temp2,10);
```

**Se** temp1 != cpf[9] **ou** temp2 != cpf[10], há erro

- Em pacotes de dados, geralmente, são adicionados bits de redundância através de um código verificador de redundância cíclica (do inglês, *cyclic redundancy check* (CRC) code)





### 3. Lidando com erros

- Em pacotes de dados, geralmente, são adicionados bits de redundância através de um código verificador de redundância cíclica (do inglês, *cyclic redundancy check (CRC) code*)

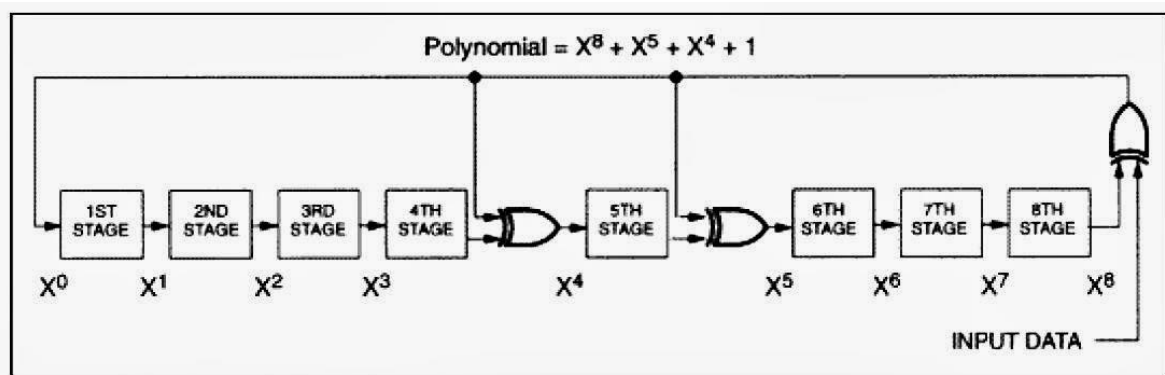


Figure 2. Maxim 1-Wire 8-bit CRC.

- O receptor ignora os dados errados, mas o transmissor não sabe disso. Como resolver essa questão?

Protocolos para comunicação confiável  
(veremos alguns rudimentos em breve)

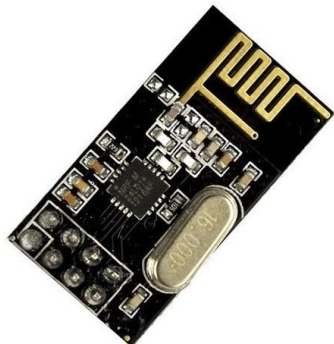


## 4. Múltiplos Usuários e Endereçamento

- No caso de múltiplos usuários, faz-se necessário endereçar o envio dos dados
- Os dados são enviados em pacotes (datagramas), *i.e.*, um conjunto de bits (ou bytes)
- No cabeçalho do pacote (*e.g.*, nos primeiros bytes do pacote) são colocadas informações que são úteis para o tratamento do mesmo
- Entre as informações pode-se colocar o endereço do remetente e de destino (se houver mais de uma possibilidade)
- O receptor que possui o endereço de destino decodifica o pacote. Caso contrário, o pacote é passado adiante, ou descartado

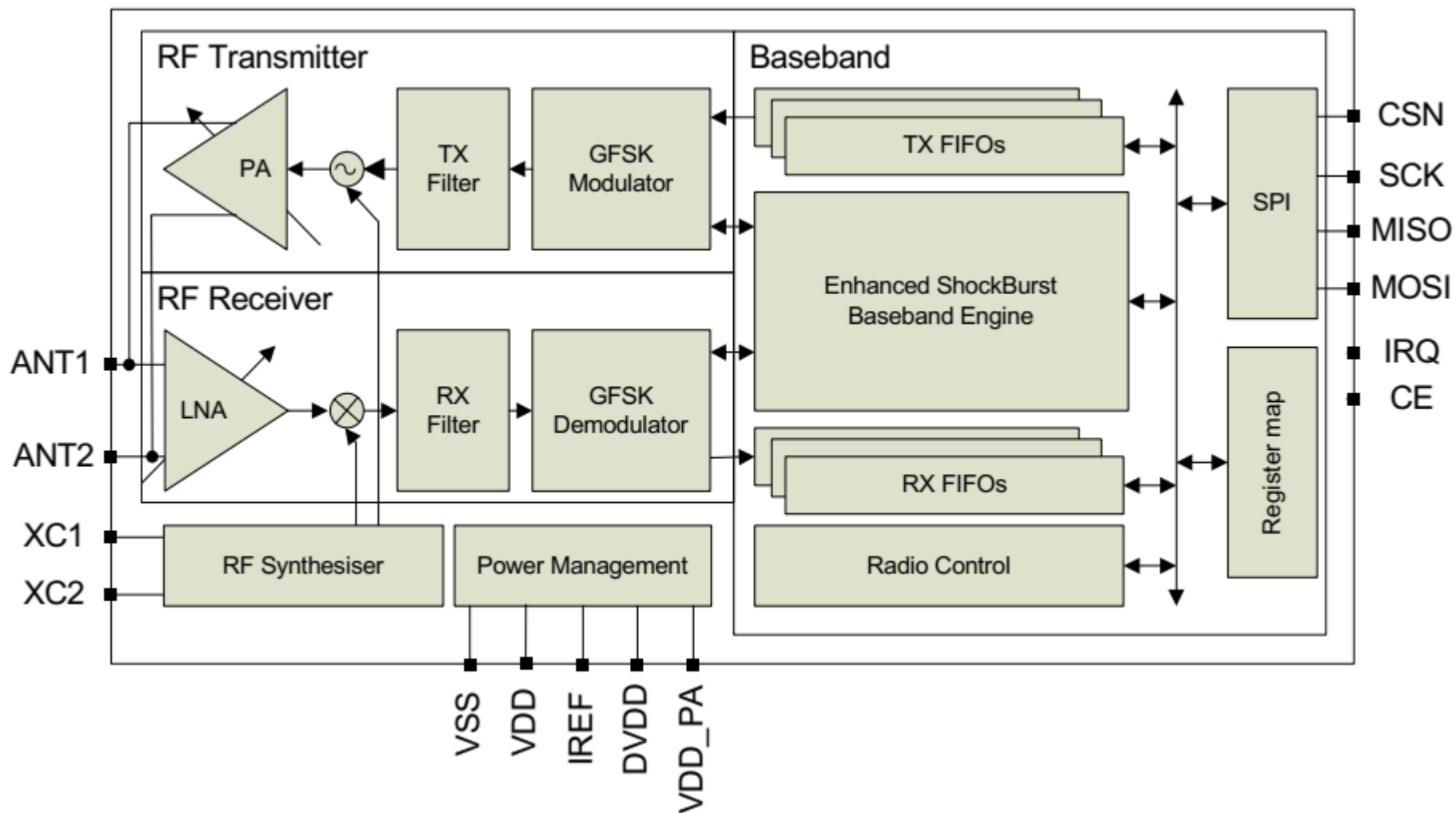
## 5. A placa NRF24L01+

- Barata (~ R\$10,00)
- Taxas de transmissão: 250 kbps a 2 Mbps
- Potências de transmissão: 15,625  $\mu$ W a 1 mW
- Alcance de 10 a 15 metros em ambientes internos (sem obstruções em ambientes externos +100m)
- Operação na banda ISM de 2,4 GHz
- Permite *multicast* e variações manuais de diversos parâmetros (e.g., potência, taxa, CRC, endereçamento, canal)
- O *driver* é simples
- Consumo máximo de 33,9 mW
- Pacotes de até 32 bytes



## 5. A placa NRF24L01+

- Diagrama de blocos





## 6. A interface *Serial Peripheral Interface* (SPI)

- É uma interface serial síncrona (diferentemente da porta serial antiga): não precisa de start e stop bits (*overhead*) e pode atingir taxas mais elevadas (até uma dezena de Mbps)
- Geralmente trabalha com a transmissão de *bytes*
- Para ser síncrono, a placa fornece o relógio: pino SCK – *Serial Clock*





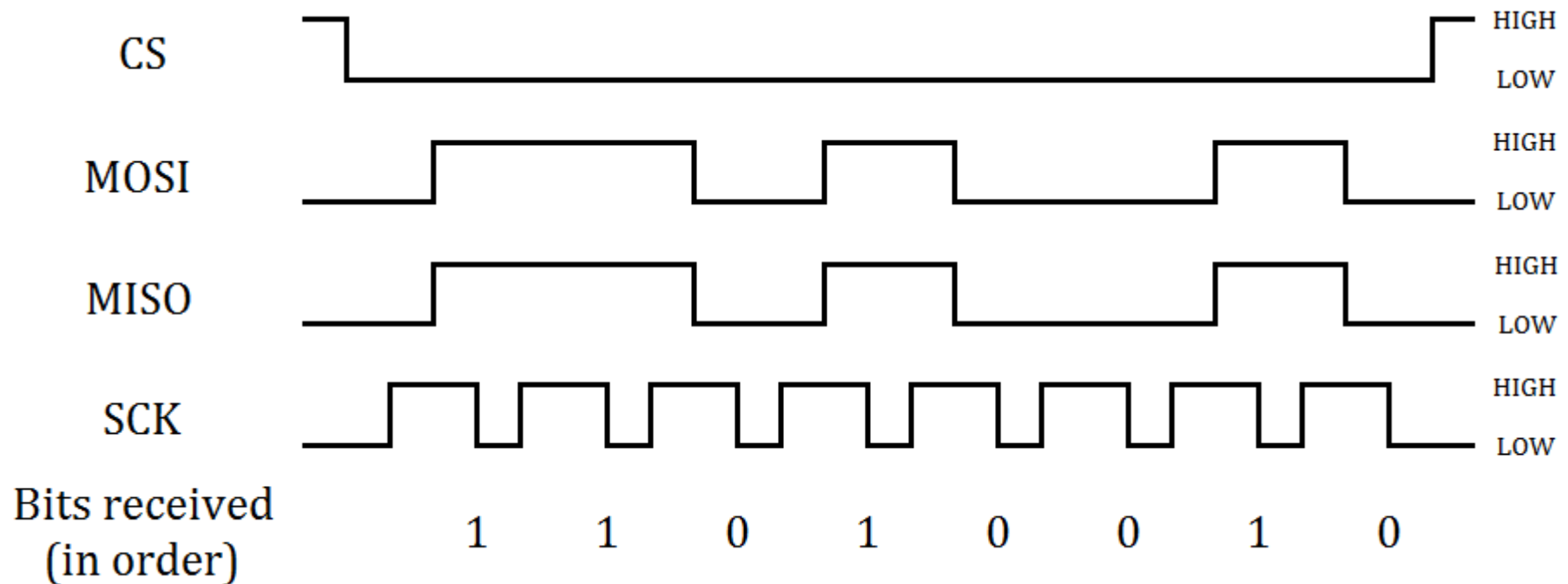
## 6. A interface *Serial Peripheral Interface* (SPI)

- Há duas vias de dados: A *Master*, que sai do micro-controlador (que gera o relógio), chamada MOSI (Master-Out Slave-In), e a *Slave*, chamada MISO (*Master-In Slave-Out*)
- O dispositivo escravo é habilitado a receber e transmitir através de um sinal *Slave Select* (SS), que é habilitado, usualmente, quando baixo
  - Na NRF24L01P, o pino se chama *Chip Select Not* (CSN)
  - O pino *Chip Enable* (CE) na NRF24L01+ é usado para determinar o modo TX e RX do RF
  - Não há pinos específicos para o SS e CE no micro-controlador – são usados quaisquer pinos GPIO
  - Pino IRQ pode ser usado para avisar de uma recepção (ao invés de fazer *pooling*)



## 6. A interface *Serial Peripheral Interface (SPI)*

- Exemplo: Envio e recepção do valor 210 (11010010) – detecção na borda de decida



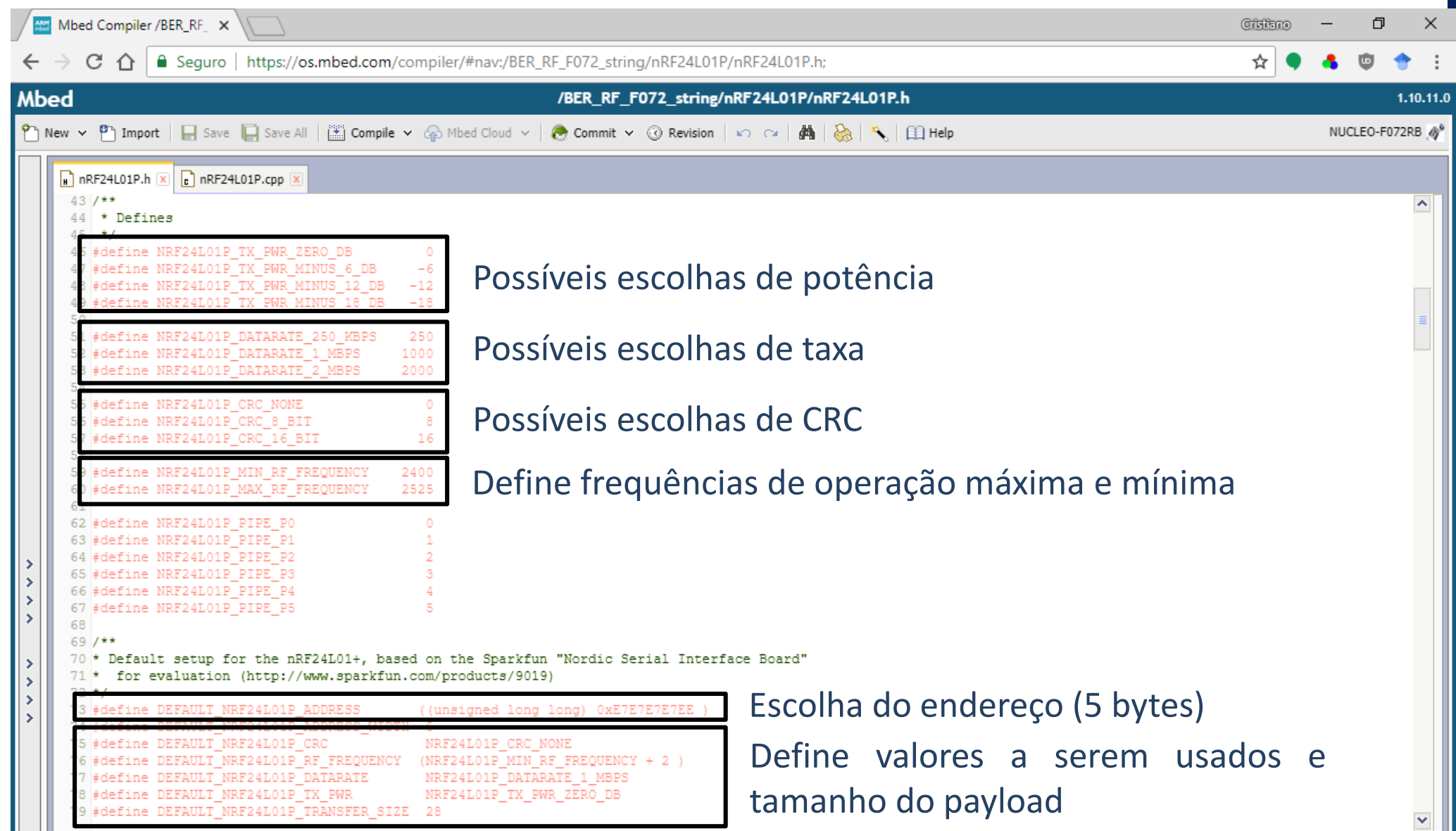
- Retirado de <https://blog.digilentinc.com/pmod-communication-serial-peripheral-interface-in-detail/>

## 7. O driver da NRF24L01+

- Há vários *drivers* disponíveis (e você pode fazer o seu seguindo o manual da NRF24L01+)
- Usaremos um *driver* para a NRF24L01+ criado por Owen Edwards e modificado por mim para limpar as *buffers* na inicialização da placa.
- O driver é simples, bem comentado, e está disponível em <http://os.mbed.com/users/cpanazio/code/nRF24L01P/> e no Moodle.
- O que devemos saber:
  - Endereçamento
  - Taxa
  - Potência
  - CRC
  - Frequência de operação
  - Tamanho do *payload*
- Estes parâmetros estão no arquivo nRF24L01P.h ou podem ser determinados no programa através de determinadas chamadas

# 7. O driver da NRF24L01+

- Estes parâmetros estão no arquivo nRF24L01P.h



The screenshot shows the Mbed IDE interface with the file `nRF24L01P.h` open. The file contains various macros for configuring the NRF24L01+ module. Several macros are highlighted with black boxes, and their functions are explained in Portuguese to the right of the code.

```
43 /**
44  * Defines
45  */
46 #define NRF24L01P_TX_PWR_ZERO_DB      0
47 #define NRF24L01P_TX_PWR_MINUS_6_DB   -6
48 #define NRF24L01P_TX_PWR_MINUS_12_DB  -12
49 #define NRF24L01P_TX_PWR_MINUS_18_DB  -18
50
51 #define NRF24L01P_DATARATE_250_KBPS    250
52 #define NRF24L01P_DATARATE_1_MBPS     1000
53 #define NRF24L01P_DATARATE_2_MBPS     2000
54
55 #define NRF24L01P_CRC_NONE             0
56 #define NRF24L01P_CRC_8_BIT            8
57 #define NRF24L01P_CRC_16_BIT           16
58
59 #define NRF24L01P_MIN_RF_FREQUENCY     2400
60 #define NRF24L01P_MAX_RF_FREQUENCY    2525
61
62 #define NRF24L01P_PIPE_P0              0
63 #define NRF24L01P_PIPE_P1              1
64 #define NRF24L01P_PIPE_P2              2
65 #define NRF24L01P_PIPE_P3              3
66 #define NRF24L01P_PIPE_P4              4
67 #define NRF24L01P_PIPE_P5              5
68
69 /**
70  * Default setup for the nRF24L01+, based on the Sparkfun "Nordic Serial Interface Board"
71  * for evaluation (http://www.sparkfun.com/products/9019)
72  */
73 #define DEFAULT_NRF24L01P_ADDRESS      ((unsigned long long) 0xE7E7E7EE )
74 #define DEFAULT_NRF24L01P_CRC           NRF24L01P_CRC_NONE
75 #define DEFAULT_NRF24L01P_RF_FREQUENCY (NRF24L01P_MIN_RF_FREQUENCY + 2 )
76 #define DEFAULT_NRF24L01P_DATARATE     NRF24L01P_DATARATE_1_MBPS
77 #define DEFAULT_NRF24L01P_TX_PWR       NRF24L01P_TX_PWR_ZERO_DB
78 #define DEFAULT_NRF24L01P_TRANSFER_SIZE 28
```

Possíveis escolhas de potência

Possíveis escolhas de taxa

Possíveis escolhas de CRC

Define frequências de operação máxima e mínima

Escolha do endereço (5 bytes)

Define valores a serem usados e tamanho do payload



## 8. Parâmetros da NRF24L01+ : Potência

- A potência pode assumir os valores: 0, -6, -12 e -18 dBm
- O que é dBm?
  - dB é usada como medida de **razão de potências**
  - Definição:  $10\log_{10}(P_2/P_1)$ , em que são as razões de potências
  - dBm é a comparação de uma potência  $P_2$  com  $P_1=1\text{mW}$
  - Assim,  $0 \text{ dBm} \Rightarrow P_2=1 \text{ mW}$ ,  $-6 \text{ dBm} \Rightarrow P_2=250 \text{ } \mu\text{W}$ ,  $-12 \text{ dBm} \Rightarrow P_2=62,5 \text{ } \mu\text{W}$  e  $-18 \text{ dBm} \Rightarrow P_2=15,625 \text{ } \mu\text{W}$
- Menor a potência, menor o consumo, mas menor o alcance





## 9. Parâmetros da NRF24L01+ : Taxa

- As taxas podem assumir os valores: 250 kbps, 1 Mbps e 2Mbps
- Por que usar diferentes valores?
  - A probabilidade de erro nos bit recebidos depende da energia em cada bit
  - Sabe-se que a Potência = Energia / Tempo = Energia x Taxa
  - Logo, dado que a *potência de transmissão é constante*, quanto *maior a taxa*, *menor a energia em cada bit*
  - Mas *taxas mais altas* permitem *transmitir mais dados em menos tempo*: isto *libera por mais tempo o canal*, que *pode ser compartilhado* e também, *menor é a chance de colisão* (i.e., sobreposição de dois ou mais sinais de usuários no receptor), o que ocasiona erros



## 10. Parâmetros da NRF24L01+ : Endereço

- A NRF24L01+ trabalha com endereços de 3 a 5 bytes tanto para a recepção (RX) como para transmissão (TX)
- Além do receptor, os endereços também atribuem qual pipeline será usado para recepção
- Por simplicidade e para trabalhar alguns conceitos, esses endereços do driver da NRF24L01+ serão fixados como um único endereço: E7E7E7E7E7h (obs.: este h é para dizer representação hexadecimal)
- Adotaremos apenas endereço do terminal remetente que será incluído no payload
  - A ideia é implementar alguns protocolos sem entrar no mérito do driver
  - Mais detalhes no último experimento deste tópico



# 11. Parâmetros da NRF24L01+ : CRC

- Códigos CRC – *Cyclic Redundancy Check*: é usado para verificar se há erros no sinal recebido
- Não garante que sempre descobrirá a presença de erros
- Quando for detectado erro, o sistema descarta o pacote
- Quanto maior o CRC, menor é a chance de não perceber erros
  - CRC-r bits por pacote de menos de  $2^{r-1}$  bits, os seguintes erros podem ser detectados:
    1. Todos os padrões de erro de 1, 2 ou 3 erros de bit
    2. Todas rajadas (sequências) de erro de r ou menos bits
    3. Um grande número de erros com probabilidade  $1-2^{-r}$
- Contudo, mais bits terão que ser transmitidos  $\Rightarrow$  maior gasto de energia e mais tempo ocupando o canal
- O CRC, na NRF24L01+ cobre todo o pacote transmitido, i.e., cabeçalho e *payload* (informação)
- As opções de CRC são: ausência (0 bit), 8 e 16 bits



## 12. Parâmetros da NRF24L01+ : Frequência de operação

- A faixa de operação é de 2400 MHz até 2525 MHz
- O driver permite escolher a frequência de operação: incrementos de 1 MHz quando a taxa é de 250kbps ou 1Mbps e incrementos de 2 MHz quando a taxa é de 2 Mbps
  - Obedecendo isto, não há sobreposição de sinais em frequências de operação (canais) distintos
- Exemplo em `nRF24L01P.h`: operação em 2402 MHz

```
#define NRF24L01P_MIN_RF_FREQUENCY      2400
#define DEFAULT_NRF24L01P_RF_FREQUENCY  (NRF24L01P_MIN_RF_FREQUENCY + 2 )
```



## 13. Parâmetros da NRF24L01+ : Tamanho do *payload*

- O *driver* e a NRF24L01+ permitem trabalhar com diferentes número de bytes no *payload*, variando de 1 até 32 bytes
- O tamanho (em bytes) é definido inicialmente pela constante `DEFAULT_NRF24L01P_TRANSFER_SIZE` em `nRF24L01P.h`
- A NRF24L01+ também tem uma opção de tamanho dinamicamente variável implementada, mas não usaremos ela a priori





## 14. Experimento 1: Demonstração Espectro vs Taxa

- Experimento/Demonstração 1:  
Densidade Espectral de Potência e Ocupação Espectral vs Taxa  
de Transmissão



## 14. Experimento 1: Demonstração Espectro vs Taxa

- O professor demonstrará como o espectro fica ocupado em relação a taxa que está sendo usada
- Será utilizado um analisador de espectro para tal demonstração




## 15. Experimento 2: Alcance vs Potência e Taxa

- Experimento 2: Alcance vs potência e taxa



## 15. Experimento 2: Alcance vs Potência e Taxa

- Conectar fios e NRF24L01+ ao micro-controlador
  - Usar oito (8) fios macho-fêmea e conectar a NRF24L01+ na sua placa micro-controladora *usando o conector padrão Arduino*
-  • Atentar que a **alimentação** é **3,3V**! Cuidado para não queimar a NRF24L01+
  - Qualquer terra (GND) do grupo POWER pode ser usado
  - Os pinos CSN, CE e IRQ podem ser colocados em qualquer porta GPIO
    - Sugestão: PA\_9, PC\_7, PA\_8



## 15. Experimento 2: Alcance vs Potência e Taxa

- Crie um novo projeto MBED (use como base o `mbed2-example-blink`), sobrescreva o conteúdo do arquivo `main.cpp` com o conteúdo do programa `exp2_aula2_receptor_NRF.cpp` (disponível no Moodle)
- Inclua a biblioteca do NRF24L01+ (disponível no Moodle ou em <http://os.mbed.com/users/cpanazio/code/nRF24L01P/>) -procure o professor em caso de dúvida em como fazer isso.
- Faça de acordo com sua conexão a declaração dos pinos na classe `my_nrf24l01p` no seu programa `main.cpp`
- Para um primeiro teste, altere os parâmetros no `nRF24L01P.h` para: 250 kbps e sem CRC (zero)
- Conecte o *Tera Term* a sua placa
- Reporte ao professor se você está recebendo
- Em seguida, o professor pedirá para serem mudados alguns parâmetros. Siga as instruções e reporte



# 16. Experimento 3: Endereçamento

- Experimento 3: Endereçamento



## 16. Experimento 3: Endereçamento

- Inspirado nos programas `exp2_aula2_receptor_NRF.cpp` e `exp2_aula2_transmissor_NRF.cpp` crie um programa que:
  - Transmita 3 bytes ao se pressionar uma determinada tecla (use a instrução `pc.getc()`, `pc` é variável definida pela classe serial, para capturar a tecla), sendo que o primeiro byte é usado para designar o endereço da sua placa
    - Use o número do seu grupo como endereço
  - Após a transmissão, você deverá receber um pacote de confirmação de sua transmissão
    - Você pode receber pacotes que não são destinados para você e estes devem ser ignorados
    - Se receber o pacote destinado a você, reproduza na tela o conteúdo do *payload*
- Use potência máxima de transmissão (0dBm) e taxa de 250 kbps





## 16. Experimento 3: Endereçamento

- **Atenção:**

- O driver permite que se fique o tempo todo no modo de recepção (`my_nrf24l01p.setReceiveMode()`), pois ele chaveia automaticamente para o modo transmissão ao usar o comando de transmissão `my_nrf24l01p.write`
- Contudo, vocês irão receber pacotes de outros terminais e estes serão armazenados em uma fila (*buffer*) da NRF24L01+
- Assim, ao transmitir um pacote, para evitar que algum lixo perturbe o processo, a fila deve ser apagada usando o comando `my_nrf24l01p.flushRx()` logo antes da transmissão