TR1: Projeto 2 - Simulação da Camada Física

Amélia Oliveira Freitas da Silva
Departamento de Ciência da Computação - CIC
Universidade de Brasília
Brasília, DF
Email: 190037971@aluno.unb.br

19 de Setembro de 2021

1 Introdução

A camada física é a responsável pela transmissão de bits (ou grupos de bits) entre agentes, permitindo a implementação de protocolos de camadas superiores.

São sua responsabilidade as seguintes operações:

- Conversão de um conjunto de informações em uma sequência interpretável de pulsos com ou sem um *clock*, a **Codificação**.
- Conversão dos pulsos digitais em sinais fisicamente transportáveis (geralmente analógicos); o que no caso de ondas portadoras, é a **Modulação**.
- O transporte físico do sinal entre os agentes.
- Conversão do sinal físico de volta aos pulsos digitais; o que no caso de ondas portadoras, é a **Demodulação**.
- Conversão dos pulsos digitais de volta à informação original, a **Decodificação**.

O objetivo da simulação é implementar a transmissão simples de bits entre agentes, que usarão a conexão para uma camada de aplicação simples (envio e mostra de bytes em ASCII).

1.1 Modulação/Demodulação

Geralmente a transmissão de sinais é realizada utilizando uma quantidade analógica variando em ondas, como uma variação de intensidade de corrente/potencial elétrico, ondas eletromagnéticas (rádio, transmissões óticas) ou outras variantes.

À onda que leva o sinal damos o nome de **Onda Portadora**. Esta onda é geralmente definida por uma senóide e possui os seguintes parâmetros:

$$O_{portadora}(t) = \alpha sin(t\omega + \theta)$$

- α: Amplitude da onda. Metade da distância entre o pico e o vale.
- ω: Velocidade angular/frequência da onda. Inverso do período, que é o tempo necessário para um ciclo completo da onda.
- θ: Fase. Deslocamento da onda no tempo. Uma fase positiva desloca a onda à esquerda.

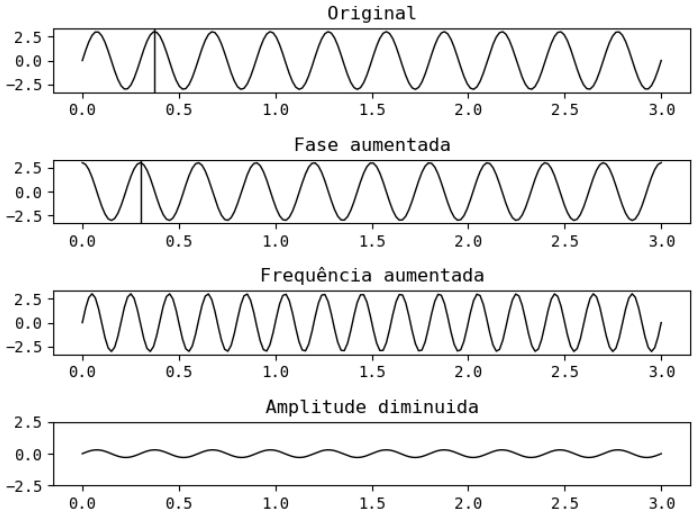


Figura 1: Ilustração dos parâmetros de onda

Alterando um ou mais dos três parâmetros no emissor podemos transmitir informações ao receptor. Essa alteração é denominada **Modulação** (como nos rádios AM - *Amplitude Modulation* - e FM - *Frequency Modulation*).

Trabalharemos com a modulação da fase para a nossa simulação. A modulação de fase é comumente utilizada para transmissões digitais em ambientes com pouco ruido por sua resistência moderada a ruídos e boa eficiência espectral.

A modulação é executada simplesmente pela alteração do parâmetro na onda portadora, enquanto a demodulação de um sinal modulado em fase é baseada na multiplicação por um sinal de referência.

$$sin(\omega t+\theta_1)sin(\omega t+\theta_2) = \frac{1}{2}[cos(\theta_1-\theta_2)-cos(2\omega t+(\theta_1+\theta_2))] \quad (1)$$

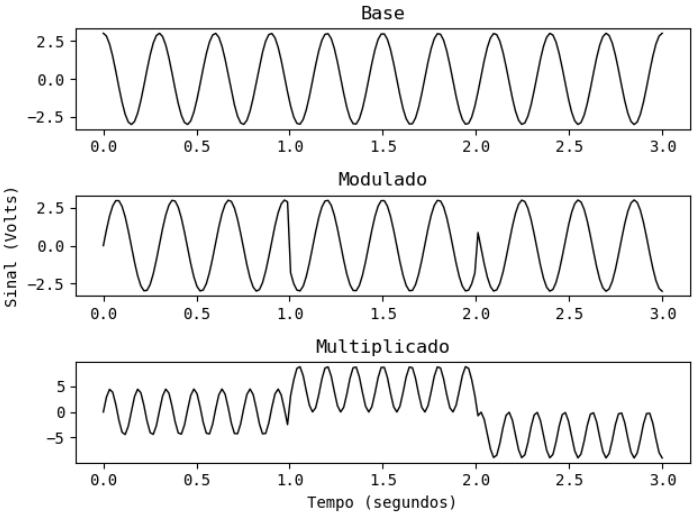


Figura 2: Ilustração da multiplicação de senoides

Pode-se ver que a onda resultante possui um deslocamento de $\frac{cos(\theta_1-\theta_2)}{2}$ em relação ao zero. Isso nos permite identificar a

diferença entre a fase do sinal e a fase de referência, ou seja, a informação por ela transmitida.

Geralmente são usadas duas fases na modulação: 0 e $\frac{\pi}{2} rad$. Na figura 2 (seção 1.1) podemos ver exemplos com modulação de $\frac{\pi}{2}$, 0 e $-\frac{\pi}{2}$, além do resultado da multiplicação dos sinais, ilustrando o deslocamento causado pela modulação.

Aplicando-se uma média móvel exponencial ao produto dos sinais, a distinção entre os três estados fica ainda melhor definida.

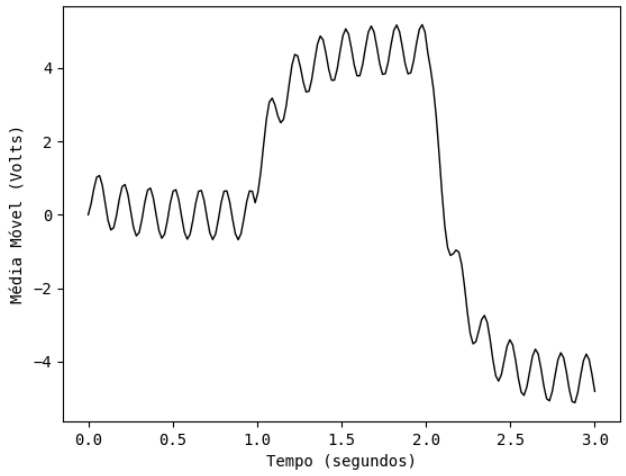


Figura 3: Média móvel exponencial do produto dos sinais

1.2 Codificação/Decodificação

Estabelecidos meios para a transmissão de pulsos e estados definidos, resta definir como a informação que transmitiremos será comunicada por meio desses estados e pulsos.

1.2.1 Codificação Binária

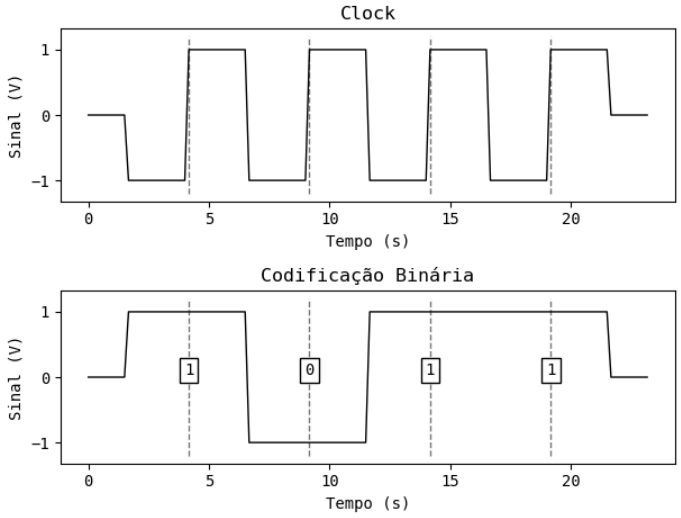


Figura 4: Demonstração da codificação binária

Na codificação binária, a representação dos bits é diretamente análoga à sequência de pulsos transmitidos.

São definidos dois valores de referência para o sinal/tensão (`ALTA`, `BAIXA`). A cada subida do *clock*, lemos o valor atual do sinal. Tensões na referência alta são geralmente lidas como "1"s e tensões na referência baixa são lidas como "0"s.

Como são usadas duas referências com valores geralmente não-nulos, a forma mais comum dessa codificação é conhecida como *NRZ* (*No Return to Zero*), dado que o sinal nunca retorna a um estado "indefinido/intermediário", permanecendo sempre em um que carrega informação.

Embora simples, essa codificação possui problemas ao representar longas cadeias de "1"s ou "0"s, pois não há transições entre as tensões altas e baixas nesses casos. Considere a figura anterior (Figura 4, 1.2.1): sem a referência do *clock*, não poderíamos saber exatamente quantos "1"s foram transmitidos entre *t=12.5s* e *t=22.5s*.

1.2.2 Codificação Manchester

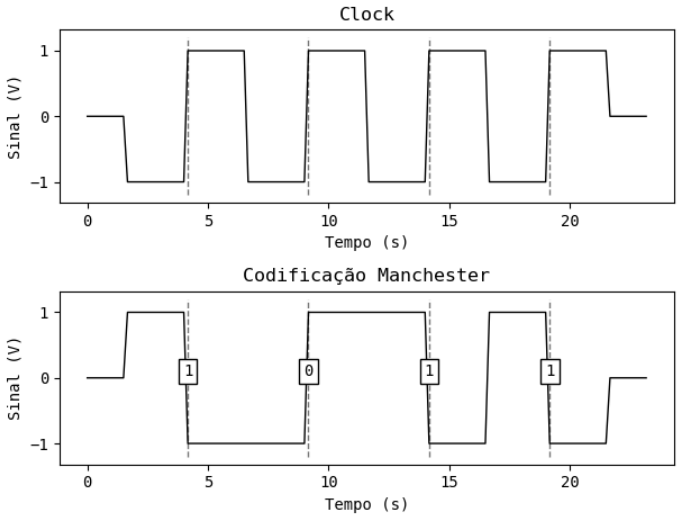


Figura 5: Demonstração da codificação Manchester

A codificação Manchester resolve o problema da falta de transições em cadeias contíguas representando "0"s e "1"s não como estados definidos, mas como transições a cada sinal de *clock*. No exemplo acima, transições "`ALTO` → `BAIXO`" representam "1"s e transições "`BAIXO` → `ALTO`", "0"s. Há transições fora do sinal de *clock* quando há bits repetidos contíguos.

A presença de transições sempre que há um sinal de *clock* permite que ajustemos a nossa referência de tempo de maneira independente. Conhecedo aproximadamente o intervalo de cada bit (período do *clock*), podemos inclusive decodificar o sinal sem a necessidade de temporização, o que torna o sinal *auto-temporizante* (*self-clocking*).

A codificação Manchester pode ser obtida como o resultado do operador binário "Ou exclusivo" entre o sinal do *clock* e a representação binária.

$$S_{manchester} = S_{binario} \otimes S_{clock}$$

1.2.3 Codificação Bipolar

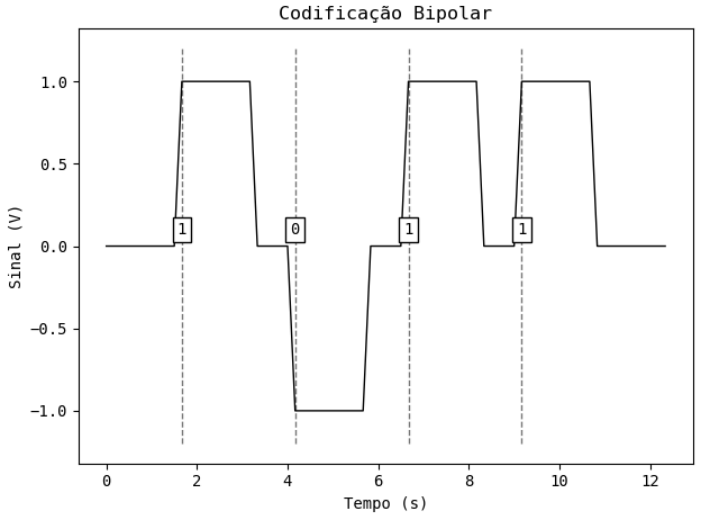


Figura 6: Demonstração da codificação Bipolar

Na codificação bipolar, ao contrário das anteriores, não necessitamos de um *clock* em ponto algum, seja na geração ou decodificação do sinal. Realizamo-no pelo uso de três valores de referência (`ALTO`,`NEUTRO`,`BAIXO`), o que a torna uma codificação *pseudoternária*.

O valor `NEUTRO` é utilizado como um separador entre os bits transmitidos. Transições "`NEUTRO`→`ALTO`"indicam "1"s, transições "`NEUTRO`→`BAIXO`"indicam "0"s (ou vice-versa a depender da especificação).

A existência de um separador de bits dispensa o uso de um *clock* externo, visto que cada transição já possui toda a informação necessária para a decodificação. Essa facilidade vem a custo da precisão necessária para a demodulação do sinal, pois precisa-se distinguir três níveis dentro do mesmo espaço de fase/amplitude/frequência, aumentando a sensibilidade do sistema a ruídos.

2 Implementação

A premissa adotada para a implementação da simulação foi definida pelos seguintes pontos:

- A simulação deve possuir parâmetros diretamente análogos a parâmetros físicos (isto é, a simulação deve ser reprodutível por um sistema físico).
- Os sistemas são causais e dinâmicos, recebendo a cada "quadro"da simulação apenas os parâmetros atuais e dependendo apenas deles (e, implicitamente, das entradas em tempos anteriores).

2.1 Modulação/Demodulação

A função `onda_base` gera o valor de uma onda portadora de amplitude 1 para um certo ponto no tempo, dados os parâmetros de frequência (implicitamente, pelo período), e fase.

As classes de modulação e demodulação possuem três estados possíveis: `enum estado = {BAIXO,NEUTRO,ALTO}`. A cada intervalo de tempo, elas atualizam seus parâmetros de acordo com a especificação seguinte e disparam os eventos respectivos.

2.1.1 Classe Modulador

A classe modulador tem uma saída analógica (`double saida`), gerada a cada intervalo de tempo baseada em seu estado atual (`estado S`), que define a fase e parâmetros de frequência, amplitude, etc.

Estado	Fase
`BAIXO`	$\frac{\pi}{2}$
`NEUTRO`	0
`ALTO`	$-\frac{\pi}{2}$

Tabela 1: Referências de fase para diferentes estados

Um intervalo de tempo é marcado pela chamada da função `Modulador::passatempo`.

O sistema é análogo à implementação física de um seletor com três saídas definidas por geradores de onda com as fases listadas e a seleção definida pelo estado atual.

2.1.2 Classe Demodulador

A classe demodulador tem uma saída discreta (`estado S`) e dispara eventos a cada mudança de estado, descrevendo transições positivas e negativas (listadas abaixo). Como entrada, ela recebe a amostra atual do sinal (saída de um modulador, atrasada ou não).

Estados	Evento
`BAIXO`→`NEUTRO`	`Demodulador::posedge`
`NEUTRO`→`ALTO`	`Demodulador::posedge`
`ALTO`→`NEUTRO`	`Demodulador::negedge`
`NEUTRO`→`BAIXO`	`Demodulador::negedge`

*Transições diretas entre `ALTO` e `BAIXO` nunca acontecem sem a passagem por um estado `NEUTRO`

Tabela 2: Eventos de transição de estado

O demodulador possui sua própria onda de referência, calculada com uma fase de referência `Demodulador::fase`.

A cada intervalo de tempo, o demodulador multiplica o sinal atual pelo sinal de referência (vide figura 2, seção 1.1), adiciona o resultado a uma média móvel exponencial (figura 3, mesma seção) e toma um dos seguintes caminhos:

- Modo **Ajuste** (função `Demodulador::ajusta_fase`): o demodulador compara o valor atual da média móvel a uma referência de valor `ALTO` e, assumindo que a entrada está em um estado `ALTO`, ajusta a fase para que a média se aproxime da referência (ou seja, alinhando os picos das ondas).
- Modo **Ativo** (função `Demodulador::recebe_amostra`): o demodulador compara o valor atual da média móvel à referência do estado atual. Caso a distância absoluta seja maior que `Demodulador::delta_max`, uma mudança de estado ocorre, alterando `Demodulador::S` e causando um dos eventos listados acima.

O modo de ajuste de fase é importante para compensar diferenças de fase entre o modulador e demodulador, que podem acontecer por falta de sincronização natural entre os parâmetros em situações reais, ou até mesmo pela distância e consequente atraso de transporte da onda entre os dois (como demonstrável na simulação).

A operação de multiplicação de sinais possui um componente físico que realiza a mesma operação, enquanto a média móvel exponencial é equivalente à implementação de um circuito RC (com a entrada passando a um capacitor por meio de um resistor e a saída sendo a diferença de potencial entre os polos do capacitor).

A taxa de alisamento σ é relacionada à capacitância C e a resistência R do circuito da seguinte forma:

$$RC = \Delta_T \left(\frac{1-\sigma}{\sigma} \right) \tag{2}$$

2.2 Codificação/Decodificação

As classes `Transmissor`(codificador) e `Receptor`(decodificador) possuem, respectivamente, dois moduladores e demoduladores representando os sinais de clock e dados. Neste nível, abstraimos a parte de modulação e focamos apenas nas transições de estados lógicos.

2.2.1 Classe Transmissor(Codificador)

O transmissor possui um *buffer* de bytes a serem transmitidos (um detalhe para facilitar a implementação da camada de aplicação, baseada em bytes, ainda que a transmissão seja bit-a-bit). Uma consequência dessa estrutura é que o transmissor só para a codificação após transmitir múltiplos de 8 bits, mas os bits não utilizados podem ser simplesmente ignorados.

Para cada bit a transmitir, o codificador passa por um ciclo definido, alterando o estado dos seus moduladores para transmitir a informação codificada. Para a codificação Manchester, foi introduzido um pequeno atraso no sinal de dados para facilitar a futura decodificação.

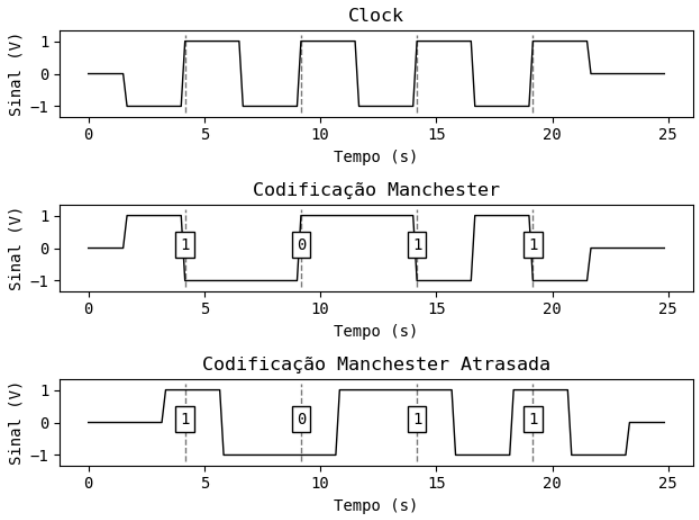


Figura 7: Codificação Manchester com atraso

Os ciclos de transmissão para cada codificação são os seguintes:

- **Codificação binária**:
  - Tempo 0: O modulador de dados vai para o estado Bit_{atual};
  - Tempo 1: O modulador de clock vai para o estado `ALTO`;
  - Tempo 2: O modulador de clock vai para o estado `BAIXO`.
- **Codificação Manchester**:
  - Tempo 0: O modulador de clock vai para o estado `BAIXO`
  - Tempo 1: O modulador de dados vai para o estado Bit_{atual};
  - Tempo 2: O modulador de clock vai para o estado `ALTO`;
  - Tempo 3: O modulador de dados vai para o estado $\overline{Bit_{atual}}$ (equivalente a $clock \otimes$ bit atual);
- **Codificação bipolar**:
  - Tempo 0: O modulador de dados vai para o estado Bit_{atual};
  - Tempo 1: O modulador de dados vai para o estado `NEUTRO`;

No código, os ciclos estão divididos em quatro passos nos três casos, com passos redundantes nos ciclos com menos de quatro.

O trânsito entre todos os passos do transmissor leva um tempo fixo e ao final de cada ciclo garantimos a codificação de um bit. A taxa de transmissão de bits é, pois, o inverso do tempo de um bit (`Transmissor::periodo_bit`).

2.2.2 Classe Receptor (Decodificador)

O receptor lê e interpreta os eventos dos demoduladores de maneira diferente a depender da codificação utilizada. Um *buffer* de 8 bits é utilizado para facilitar a comunicação com a aplicação, causando um evento (`Receptor::on_char`) quando os 8 bits são preenchidos.

Para cada codificação, os eventos são interpretados de acordo com as definições a seguir:

Estado (Clock)	Ação Binária
`BAIXO`→`NEUTRO`	N/A
`NEUTRO`→`ALTO`	Atribui o valor atual do sinal ao bit
`ALTO`→`NEUTRO`	N/A
`NEUTRO`→`BAIXO`	N/A

Tabela 3: Transições para a codificação binária

Estado (Clock)	Ação Manchester
`BAIXO`→`NEUTRO`	N/A
`NEUTRO`→`ALTO`	Atribui o valor atual do sinal ao bit
`ALTO`→`NEUTRO`	N/A
`NEUTRO`→`BAIXO`	N/A

Tabela 4: Transições para a codificação Manchester (Atrasada)

Estado (Dado)	Ação Bipolar
`BAIXO`→`NEUTRO`	N/A
`NEUTRO`→`ALTO`	$Bit_{atual} = 1$
`ALTO`→`NEUTRO`	N/A
`NEUTRO`→`BAIXO`	$Bit_{atual} = 0$

Tabela 5: Transições para a codificação Bipolar

Como atrasamos a codificação manchester, na transição do clock para o estado `ALTO`, o valor do sinal ainda não foi atualizado de

$$Bit_{transmitido} \oplus 0 = Bit_{transmitido}$$

para

$$Bit_{transmitido} \oplus 1 = \overline{Bit_{transmitido}}$$

(O que aconteceria no próximo passo do ciclo de transmissão)

Podemos então simplesmente lê-lo diretamente, como na binária, simplificando bastante a decodificação.

A cada definição de bit, passamos para o próximo até preencher os 8, voltando ao primeiro depois do disparo do evento.

2.3 Aplicação

Na camada de aplicação, abstraimos a codificação/decodificação dos bits (e empacotamento em bytes), então trabalharemos somente com a interface do usuário e em dar significados aos bytes transmitidos.

A aplicação é baseada na transmissão de letras no padrão ASCII de um agente a outro por meio do sistema até agora descrito. Cada tecla digitada é enviada como um byte ao transmissor, decodificada no receptor e adicionada a um *buffer* para visualização.

Algumas teclas possuem funções especiais e não serão transmitidas, mas sim causarão eventos específicos ao serem pressionadas:

Tecla	Efeito
`1`	Usa a codificação binária
`2`	Usa a codificação Manchester
`3`	Usa a codificação bipolar
`ESPAÇO`	Redefine a simulação

Tabela 6: Teclas-chave na aplicação final

O transporte da onda entre os agentes é mediado por uma linha de transmissão, o que introduz um atraso entre o envio e recebimento da informação, causando um desvio de fase. A linha é determinada diretamente pela largura da tela, o que permite testar facilmente o impacto de mudanças do tamanho da linha.

Para a mostra da informação na tela foi usada a biblioteca `ncurses` de interfaces baseadas em texto.

3 Membros

Não aplicável (trabalho individual).

4 Conclusão

Os objetivos do projeto foram considerados cumpridos e todos os critérios de avaliação foram levados em conta no desenvolvimento.

Estou satisfeita com o resultado final do projeto, especialmente após a maximização da taxa de transmissão.

As dificuldades encontradas foram as seguintes:

- Testes com os moduladores são extremamente lentos com parâmetros normais (a taxa original de transmissão era de 0.2 bit/s, posteriormente otimizada para 0.8 bit/s);
- Os moduladores são extremamente sensíveis a mudanças de parâmetros quando necessitamos transições mais rápidas a fim de acelerar a transmissão e testes;
- A determinação dos parâmetros ótimos foi feita por tentativa-e-erro utilizando medições do máximo da média móvel e determinação visual da estabilidade de cada estado, complicando o ajuste final;
- O uso da biblioteca `ncurses` foi ineficiente, causando problemas com certas alturas de janelas.
  - Janelas com altura menor que 30 caracteres omitem as ilustrações do ponteiro de bits;
  - Janelas com largura menor que 90 caracteres omitem parte da informação dos agentes;
  - Janelas cuja altura resulta em resto 1 quando dividida por três omitem o sinal de dados por completo.

Propostas de desenvolvimentos futuros incluem:

- Automatização da escolha dos parâmetros para o modulador: determinada a fase correta, encontrando-se os máximos e mínimos de cada estado podemos automatizar a escolha dos parâmetros `Demodulador::referencia` e `Demodulador::delta_max`.
- Automatização da escolha de taxa de transferência: há uma determinada taxa máxima teórica de transferência para uma onda. A taxa máxima será menor na prática devido a erros de aproximação, pequenas diferenças de fase ignoradas pelo ajuste e outros fatores. Determinar essa taxa nos permite tornar a transferência mais rápida, permitindo a implementação de camadas mais complexas de aplicação.

Todos os gráficos-exemplo foram gerados em python usando a antiga taxa de transferência de 0.2 bit/s.

Agradeço pelo tempo e leitura.

APÊNDICE

Implementando-se o circuito de modulação/demodulação em um simulador físico podemos realizar análises espectrais da transmissão final.

O circuito de modulação foi implementado como um seletor analógico entre três geradores de onda, ou seja, uma série de multiplicações e adições implementando a seguinte fórmula (assumindo que o sinal varia entre 0 a 1):

$$\begin{aligned} S_{modulado} = Onda_{baixa} * (\overline{S_{alto}} * \overline{S_{neutro}}) \\ +Onda_{alta} * (S_{alto} * \overline{S_{neutro}}) \\ +Onda_{neutra} * S_{neutro} \end{aligned} \quad (3)$$

O circuito de demodulação consiste em um gerador de onda na referência de fase `ALTA` conectado a um multiplicador que recebe o sinal de entrada. A saída do multiplicador é conectada por um resistor à saída, que é conectada à referência aterrada da onda por um capacitor.

O circuito após a multiplicação (responsável por "alisar" o sinal) também é conhecido como "filtro passa-baixas RC de primeira ordem".

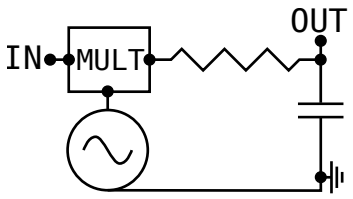
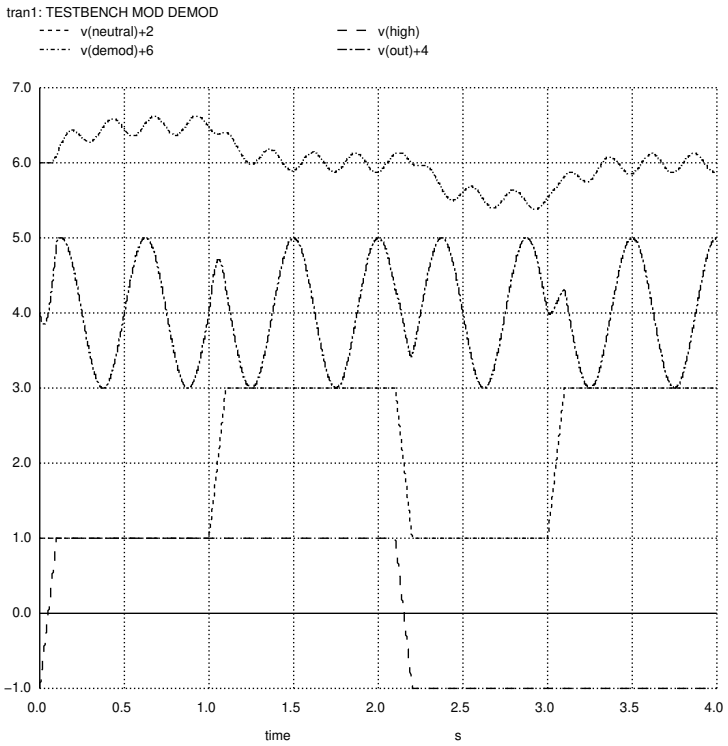


Figura 8: Circuito demodulador

Os circuitos foram implementados na especificação de simulação de circuitos/sinais mistos `NGSPICE`.

A figura seguinte demonstra as funcionalidades dos circuitos de modulação e demodulação, mostrando os resultados para as quatro combinações possíveis das duas entradas digitais (S_{neutro} e S_{alto}). Foram usados um resistor de 1.5Ω e um capacitor de 100mF.



Em ordem:

- Sinal demodulado
- Sinal modulado
- S_{neutro}
- S_{alto}

Figura 9: Resultado da simulação física

Para a análise espectral, foi simulada uma sequência de 10 segundos de bits alternando-se entre 0 e 1 e aplicada a transformada rápida de Fourier nos sinais resultantes.

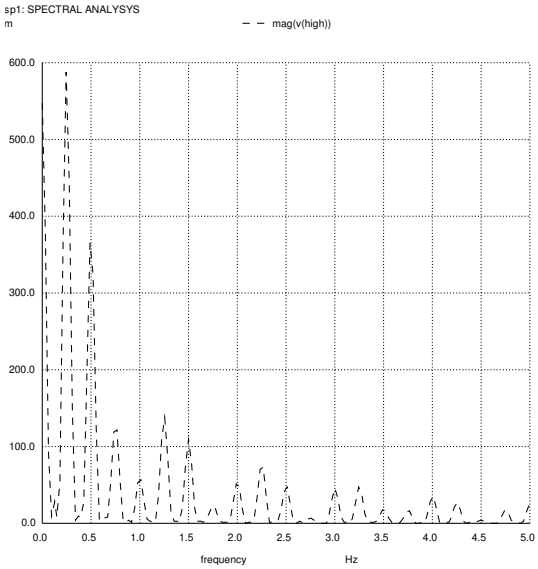


Figura 10: Análise espectral para o NRZ (sinal "high")

Para a codificação NRZ (equivalente ao nosso sinal "high"no modulador), percebemos a esperada curva com os harmônicos

baixos (e a constante - f=0) contribuindo com quase toda a energia do sinal.

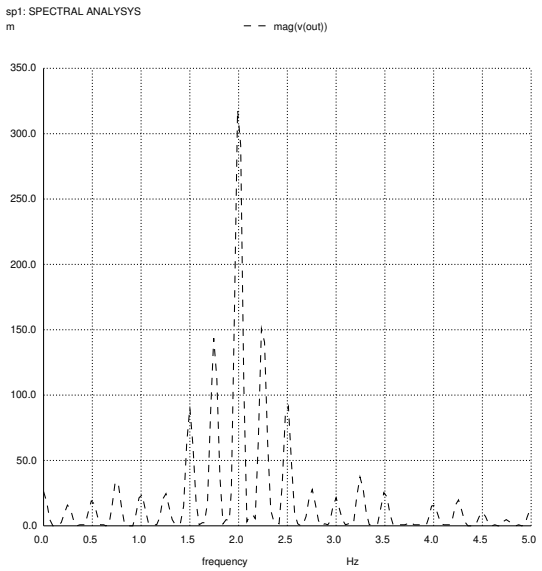


Figura 11: Análise espectral para o sinal modulado

Com a modulação, as componentes que carregam a maior energia se concentram ao redor da frequência da onda portadora, garantindo que a energia está sendo usada na maior parte apenas para a transmissão de informações, além de permitir que sintonizemos o nosso receptor automaticamente ao buscar o pico principal.

Todo o código aqui utilizado - inclusive o fonte deste relatório - está disponível no repositório *https://github.com/m3101/TR1-P2*, que será aberto ao público ao final do prazo do projeto