Universidade de Brasília
Departamento da Ciência da Computação
Teleinformática e Redes 1
Dr. Marcelo Antônio Marotta
Alícia Rita Oliveira dos Reis – 17/0129306
Cintia Leal Rodrigues – 17/0125696

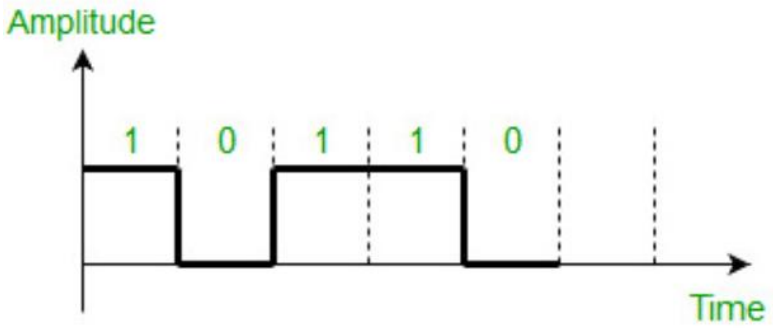
Camada Física

Brasília, 03/04/2022

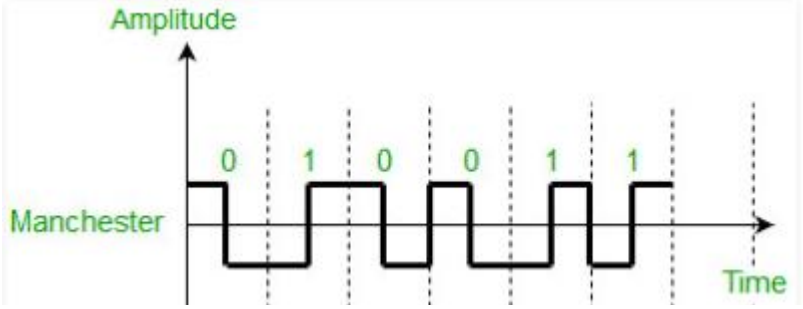
Introdução

A camada física, lida com a transmissão em nível de bits entre diferentes dispositivos e suporta interfaces elétricas ou mecânicas que se conectam ao meio físico para comunicação sincronizada, criando um sinal elétrico, óptico ou micro-ondas que representa os bits. Esta camada joga com a maioria das conexões físicas da rede, transmissão sem fio, cabeamento, padrões e tipos de cabeamento, conectores e tipos, placas de interface de rede e muito mais, de acordo com os requisitos de rede.

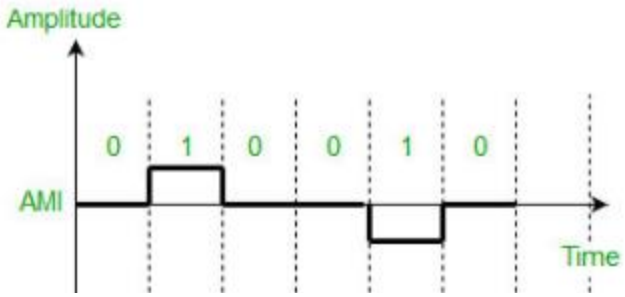
O protocolo Binário, é feita a tradução no dispositivo que em definição o bit 1 a tensão é positiva e quando o bit é 0 a tensão é zero, mas o sinal não retorna zero no meio do bit por isso é chamado NRZ. Utilizando o exemplo a seguir para demostrar o sinal de tensão.



O protocolo Manchester, combina RZ com NRZ-L, deste modo a transição do bit é dividida em duas partes. Assim, o nível permanece em um nível durante a primeira metade e se move para o outro nível na segunda metade, realizando a sincronização de transição no meio do bit.



O protocolo Bipolar, para este caso a tradução no dispositivo é feita em três níveis positivo, negativo e zero. Se tensão for positiva ou negativa ela ficará se alternando, enquanto se a tensão for zero o nível é igualmente nulo. Utilizando o exemplo a seguir para demostrar o sinal de tensão.



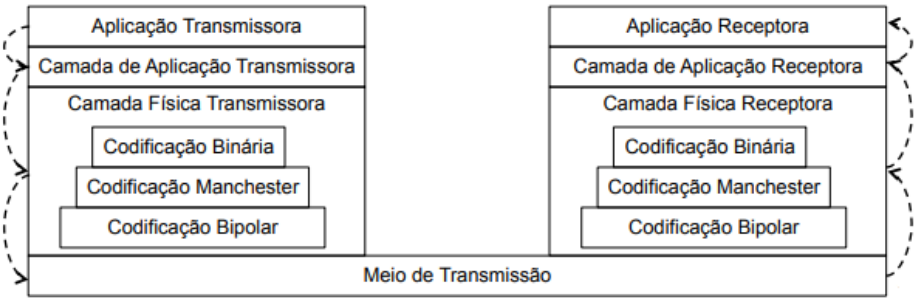
Implementação

Para a implementação foram criados três arquivos camadaFisica.cpp, camadaFisica.hpp, simulador.cpp e um makefile. O simulador é composto pela camada de aplicação. O trabalho foi feito no VScode e compilado no WSL. Para executar o programa basta abrir o terminal e digitar “make”, onde é gerado o código objeto, “make run” para executar e “make clean” para limpar o diretório.

Após compilar com o makefile, é solicitado uma mensagem para simular o funcionamento da camada física, ao escrever sua mensagem o programa transforma a mensagem inserida em um código ASCII.

O usuário deve selecionar o tipo de protocolo entre binário, manchester e bipolar. A mensagem é enviada para a camada de aplicação transmissora onde é codificada de acordo com a tabela ASCII, esse código é enviado para a camada física transmissora onde será escolhida o tipo de protocolo.

A mensagem codificada é enviada para o meio de transmissão, que é responsável pelo canal de comunicação entre a transmissão e recepção do sinal. O sinal é decodificado na camada física receptora e enviado para a camada de aplicação receptora que vai transformar o código em uma mensagem novamente, essa mensagem será exibida na aplicação receptora. A imagem abaixo mostra a simulação do projeto.



Protocolo Binário:

Para o protocolo binário a mensagem foi transformada em código ASCII, como a mensagem já foi codificada em binário para entrar na camada física transmissora, não houve modificação. O mesmo se aplica na camada física receptora, como não houve mudança no vetor anterior não é preciso mudar para a decodificação.

O vetor quadro vai printar cada elemento do vetor binário, de acordo com o tamanho do vetor.

```
/* Binária */
vector<int> camadaFisicaTransmissora_Codificacao_Binaria(vector<int> quadro) {
  cout << endl << "Codificação Binária:" << endl;
  unsigned tam = quadro.size(), i;
  for (i = 0; i < tam; i++) {
    cout << quadro.at(i); /* printa o elemento do vetor em codigo ASCII*/
    cout << " ";
  }
  cout << endl;
  return quadro;
}

vector<int> camadaFisicaReceptora_Codificacao_Binaria(vector<int> fluxoBrutoDeBits) {
  return fluxoBrutoDeBits;  /* retorna o fluxo bruto de bits */
}
```

Para teste, usamos como exemplo a mensagem “Tr1”, no terminal do WSL, podemos verificar que o resultado foi o que já era esperado:

```
cintialr@CLR:/mnt/c/Users/cinti/OneDrive/Documentos/GitHub/Camada_Fisica_TR1$ make run
./simulador
Digite a mensagem: Tr1
Selecione o tipo de protocolo:
        1 -> Binário
        2 -> Manchester
        3 -> Bipolar
1

Codificação Binária:
0 1 0 1 0 1 0 0 0 1 1 1 0 0 1 0 0 0 1 1 0 0 0 1

Mensagem recebida: Tr1
```

Protocolo Manchester:

Para esse protocolo foi utilizada a operação XOR, para percorrer o vetor do sinal recebido, assim podemos aplicar a operação em cada bit, com os bits de zero e as batidas do clock. Na parte decodificação foi realizado um loop inteirando o sinal que foi codificado em partes de dois em dois, terminando apenas quando a mensagem é decodificada.

```
/* Manchester */
vector<int> camadaFisicaTransmissora_Codificacao_Manchester(vector<int> quadro) {
  cout << endl << "Codificação Manchester:" << endl;
  vector<int> clock{0, 1};    /* representa o clock */
  vector<int> bits;
  unsigned tam = quadro.size(), i;
  for (i = 0; i < tam; i++) { /* quadro XOR clock */
    bits.push_back(quadro.at(i) ^ clock.at(0));   /* adiciona (quadro XOR clock(0)) */
    bits.push_back(quadro.at(i) ^ clock.at(1));   /* adiciona (quadro XOR clock(1)) */
    if(i % 8 == 0)
      cout << endl;

    /* printa (quadro XOR clock) */
    cout << (quadro.at(i) ^ clock.at(0)); /*at -> elemento de acesso*/
    cout << " ";
    cout << (quadro.at(i) ^ clock.at(1));
    cout << " ";
  }
  cout << endl;
  return bits;
}


vector<int> camadaFisicaReceptora_Codificacao_Manchester(vector<int> fluxoBrutoDeBits){
  vector<int> bits;
  unsigned tam = fluxoBrutoDeBits.size(), i;
  for (i = 0; i < tam; i += 2) {  /* busca de 2 em 2*/
    bits.push_back(fluxoBrutoDeBits.at(i));  /* recuperação do sinal original */
  }
  return bits;
}
```

Para teste, usamos como exemplo a mensagem “Tr1”, no terminal do wsl, podemos verificar que o resultado foi o que já era esperado:

```
cintialr@CLR:/mnt/c/Users/cinti/Downloads/Camada_Fisica_TR1-main (1)/Camada_Fisica_TR1-main$ make run
./simulador
Digite a mensagem:
Tr1
Selecione o tipo de protocolo:
        1 -> Binário
        2 -> Manchester
        3 -> Bipolar
2

Codificação Manchester:
0 1 1 0 0 1 1 0 0 1 1 0 0 1 0 1 0 1 1 0 1 0 1 0 0 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 0 1 1 0

Mensagem recebida: Tr1
```

Protocolo Bipolar:

Protocolo bipolar foi utilizado assim como no protocolo binário, aqui é verificado se o bit for é igual a zero não muda, se o bit for igual a um, o sinal muda de polo, obtendo um polo positivo e outro negativo. Para a decodificação fazemos o modulo para cada bit, assim podemos recuperar os bits negativos, retornando a mensagem originalmente transmitida.

```
/* Bipolar */
vector<int> camadaFisicaTransmissora_Codificacao_Bipolar(vector<int> quadro){
  cout << endl << "Codificação Bipolar:" << endl;
  bool bipolar = false;  /*  O bipolar começa com 1 positivo */
  unsigned tam = quadro.size(), i;
  for (i = 0; i < tam; i++){
    if(quadro.at(i) == 0){  /* caso for 0, não muda */
      cout << " 0 ";
    }
    else if(quadro.at(i) == 1){   /* caso for 1*/
      if (bipolar) {
        quadro.at(i) = -1;  /* muda o polo para -V */
        cout << "-1 ";
      } else {
        quadro.at(i) = 1;    /* muda o polo para +V */
        cout << " 1 ";
      }
     bipolar = !bipolar;
    }
  }
  cout << endl;
  return quadro;
}

vector<int> camadaFisicaReceptora_Codificacao_Bipolar(vector<int> fluxoBrutoDeBits) {
  unsigned tam = fluxoBrutoDeBits.size(), i;
  for (i = 0; i < tam; i++) {
    /* função abs -> pega o valor absoluto */
    /* função at -> acessa o elemento */
    fluxoBrutoDeBits.at(i) = abs(fluxoBrutoDeBits.at(i)); /* faz o módulo */
  }
  return fluxoBrutoDeBits;
}
```

Para teste, usamos como exemplo a mensagem “Tr1”, no terminal do wsl, podemos verificar que o resultado foi o que já era esperado:

```
cintialr@CLR:/mnt/c/Users/cinti/OneDrive/Documentos/GitHub/Camada_Fisica_TR1$ make run
./simulador
Digite a mensagem: Tr1
Selecione o tipo de protocolo:
        1 -> Binário
        2 -> Manchester
        3 -> Bipolar
3

Codificação Bipolar:
 0  1  0 -1  0  1  0  0  0 -1  1 -1  0  0  1  0  0  0 -1  1  0  0  0 -1

Mensagem recebida: Tr1
```

Membros

Alícia Rita Oliveira dos Reis: Implementação do código e realização da montagem do documento PDF.

Cintia Leal Rodrigues: Implementação do código e realização da montagem do documento PDF.

Conclusão

Portanto, ao realizarmos testes, o programa mostrou-se eficaz e cumpre fielmente aquilo que foi solicitado, pois ele recebe a mensagem codificada e a decodifica e por fim devolve a mesma sem nenhum tipo de perda de sinal. Assim, foi possível aplicarmos e praticar, vários pontos da disciplina, principalmente a linguagem de programação e conhecimento do funcionamento da camada física.

Bibliografia

https://www-geeksforgeeks-org.translate.goog/difference-between-unipolar-polar-and-bipolar-line-coding-schemes/?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=op,sc

Repositório no Github

https://github.com/aliciarita/Camada_Fisica_TR1