Universidade Federal do Rio Grande do Norte
Centro de Tecnologia - CT

Mini SDR implementado na linguagem C
ELE1717 - Grupo 3 - Problema 5 - Implementação

Albertho Síziney Costa
Isaac de Lyra Junior
Lucas Augusto Maciel da Silva
Lucas Batista da Fonseca
Sthefania Fernandes Silva

Natal, 17 de agosto de 2021

Resumo

O presente relatório tem por objetivo elaborar a implementação do projeto para um mini Software Defined Radio na disciplina de Sistemas Digitais. Tal dispositivo é capaz de modular 4 técnicas como AM, FM, ASK e a FSK. A partir do tipo de modulação selecionada pelo usuário, um display LCD será implementado para fazer a interface homem-máquina. Além disso, para estabelecer o processo de modulação de um determinado sinal, deverá ser configurado o modo de modulação e a frequência da portadora através de um potênciometro. No que tange o sistema de comunicação, será utilizado um R2R no intuito de realizar o tratamento da conversão dos sinais digitais de entrada para uma saída analógica. Com isso, serão comentados os pontos modificados e acrescentados durante a implementação, bem como os resultados obtidos durante as simulações feitas.

Palavras-Chave: Modulação de frequência, AM, FM, ASK, FSK, R2R.

Lista de Imagens

Sumário

1 Implementação

O projeto pede que o mini SDR (*Software Defined Radio*) seja simulado através de um código de programação escrito na linguagem C, implementado no microcontrolador ATMega328p. Para isso, foram utilizados os *softwares* Proteus e Microchip Studio.

1.1 Correções do projeto

1.1.1 Fórmulas de Modulação

Nem todas as fórmulas indicadas no projeto permitiram que a modulação ocorresse adequadamente. Para isso algumas alterações foram feitas.

Começando com a modulação AM, a fórmula que o grupo nos forneceu é a mostrada na equação 1.1, onde A_c é a amplitude da portadora e f_c é a frequência da portadora.

$$v(t) = Ac * w(t)cos(2\pi * f_c * t) + Ac * cos(2\pi * f_c * t) \tag{1.1}$$

A alteração da fórmula consistiu em incluir o *offset* e multiplicar o sinal de entrada por um coeficiente que melhora o envelopamento da modulação, esta será mostrada mais adiante.

Já a fórmula da modulação ASK foi alterada para utilizar a mesma equação da modulação AM, isso porque não foi possível identificar um valor coerente para o coeficiente "E_b" que a fórmula anterior possuía (equação 1.2) e notou-se que a saída com a nova fórmula se comportava como um sinal modulado por chaveamento por amplitude.

$$s(t) = \sqrt{\frac{2E_b}{T_b}} * cos(2\pi * f_c * t) \tag{1.2}$$

A fórmula da modulação FSK possui uma frequência f_1 quando o sinal de entrada está em nível lógico alto e uma frequência f_2 quando em nível lógico baixo. Ficou determinado que f_1 seria o dobro da frequência da portadora, enquanto f_2 seria a própria frequência da portadora. Além disso, de forma semelhante ao ASK, a equação antiga foi alterada para não ser preciso usar o E_b.

1.2 Programa

O sistema possui um fluxograma que contém três etapas possíveis: RUN, MODULAÇÃO e PORTADORA. Cada etapa possui funções específicas que interagem com o

usuário de maneiras distintas. Cada etapa se comunica entre si, de forma que o usuário pode ir para qualquer uma das três através dos *pushbuttons* pré definidos. Para a interação com o *display* LCD, foi criada uma biblioteca com todas as funções usadas, com o intento de reduzir o código principal. Serão abordadas cada função implementada, bem como os procedimentos realizados durante a utilização delas. A explicação de cada função pode ser vista no tópico Biblioteca do *Display*.

1.2.1 Processo MODULAÇÃO

Para que o usuário tenha acesso ao processo de modulação, deve ser conferido que foi pressionado o botão "but_m" conectado na porta PC4, ou se a *flag* "flag_m" está em nível lógico alto. A *flag* citada está sendo iniciada com valor 0 no início do código, e altera para 1 toda vez que o botão "but_m" é pressionado. Essa *flag* foi criada na intenção de assegurar que o sistema volte para a etapa Modulação sempre que seu respectivo *pushbutton* for pressionado, não importando onde o sistema esteja. A Figura 1 mostra como é aferida a entrada nos laços que implementam o processo de Modulação.

Figura 1 – Condição de Entrada Para Modulação

```
//ESTADO MODULACAO
if((PINC & 1 << 4) |  flag_m == 1 ) // Se botao M foi pressionado
{
    flag_m == 0;
    while(!(PINC & 1 << 3) && !(PINC & 1 << 2))
    {
        ADCSRA |= 1 << ADSC;
        valor_ADC = ADC;

        while(!(PINC & 1 << 3) && !(PINC & 1 << 2) && (valor_ADC <= 255)) // ESCOLHA DO MODO AM
```

Fonte: Autores.

Note que para assegurar a permanência no laço de Modulação, está sendo averiguado que os demais botões não estão sendo pressionados, sendo que no laço mais interno, também existe a segurança de que o valor lido no ADC é menor que 255. O valor lido no conversor AD vai de 0 a 1023, e como o processo de Modulação confere quatro possíveis *templates* de *displays* diferentes - onde o potenciômetro define o tipo de modulação ao rotacioná-lo - para um valor convertido de 0 a 255, o *display* mostrará o *template* que confere o modo AM.

Ao entrar no modo de seleção do AM, a variável "flag_AM" vai para alto, e as demais flags das outras modulações recebem sinal lógico baixo. Dessa forma, é possível conferir através delas qual o último tipo selecionado pelo usuário, podendo recuperar essa informação a qualquer momento do sistema. Veja a figura 2.

Após acionar as *flag*s, o cursor é apagado (não fica mais visível) através dessa função "limpa_cursor" e são chamadas duas funções: "set_cell" e "set_str". A primeira define ao LCD, em qual a célula o cursor deverá se posicionar e a segunda confere que seja

Figura 2 – Estapa de Escolha do Modo AM

```
while(!(PINC & 1 << 3) && !(PINC & 1 << 2) && (valor_ADC <= 255)) // ESCOLHA DO MODO AM
{
    flag_AM = 1;
    flag_FM = 0;
    flag_ASK = 0;
    flag_FSK = 0;
    limpa_cursor (&PORTB);
    set_cell(&PORTB, 0);
    set_str(&PORTB, "Mod: AM  F:---Hz");
    //Posicionar-se na 2 lin 1 col
    pula_linha(&PORTB);
    set_str(&PORTB, "Msg:---         ");
    ADCSRA |= 1 << ADSC;
    valor_ADC = ADC;
    //Aciona a flag caso seja pressionado algum botao
    if(PINC & 1 << 3)
    flag_p = 1;
    else if(PINC & 1 << 2)
    flag_r = 1;
}
_delay_ms(5);
ADCSRA |= 1 << ADSC;
valor_ADC = ADC;
```

Fonte: Autores.

escrita uma *string* no *display*. Note que logo após, é chamada uma função "pula_linha" cuja intenção é exatamente ir para a segunda linha e permitir que as próximas escrituras sejam feitas nela. Após chamar novamente uma função de escrita, o valor do conversor AD é requisitado novamente, com a intenção de atualizá-lo e averiguar se este ainda está abaixo de 255, caso contrário, ele irá finalizar o laço e mudar para outro tipo de modulação de acordo com o seu valor. Logo, os outros laços possuem modificação apenas nas condições de permanência do laço *while*, e na *string* enviada ao LCD. A *string* varia modificando apenas as letras que indicam se a modulação é do tipo AM, FM, ASK ou FSK.

1.2.2 Processo PORTADORA

Durante o processo "PORTADORA" o usuário deve selecionar a frequência que a onda portadora deve ter para modular a entrada. Logo, o *display* deve exibir o tipo de modulação escolhida anteriormente, exibindo também o valor que está sendo lido e convertido do potenciômetro para definir a frequência. A conversão do valor lido do ADC para a frequência escolhida, ocorre de modo que o valor exibido no *display* percorre de 100Hz a 999Hz como requisitado no projeto. Observe na figura 3, o cálculo feito para essa função, considerando "ADC" como o valor lido do conversor AD (0 a 1023) e "SAIDA" o valor a ser mostrado no display (100 a 999). A função criada para a realização dessa operação se chama "mapdec".

Como pode ser visto na figura 4, para que seja possível acessar o processo "PORTADORA" a ideia é semelhante ao processo de escolha da modulação. O usuário deverá

Figura 3 – Cálculo Para Conversão da leitura da Frequência Portadora

```
1023(ADC) - 999 (SAIDA)
0   (ADC) - 100 (SAIDA)

ADC-0    SAIDA-100      ADC    SAIDA-100
------ = ---------   => ---- = ---------
1023-0   999-100        1023      899


SAIDA = (ADC*899/1023)+100
```

Fonte: Autores.

pressionar o *pushbutton* "p_but" estando em quaisquer processos do sistema, e a partir daí, o primeiro laço *while* se responsabiliza por verificar se outro botão foi pressionado para garantir a finalização do processo da escolha da frequência portadora. Note ainda, que o laço seguinte assegura a entrada se a *flag* "flag_AM" está em nível lógico alto, logo, esse laço está realizando a escolha da frequência para uma modulação AM.

Figura 4 – Condição de Entrada Para Portadora

```
//ESTADO PORTADORA
if((PINC & 1 << 3) |  flag_p == 1 ) // Se bot◆o P foi pressionado
{
    flag_p = 0;

    PORTB &= 0x3F; //Apaga os leds
    PORTB |= 1 << PORTB6; // Acendo o led vermelho

    while(!(PINC & 1 << 4) && !(PINC & 1 << 2))
    {
        while(!(PINC & 1 << 4) && !(PINC & 1 << 2) && (flag_AM == 1)) // ESCOLHA DA FREQ (AM)
        {
```

Fonte: Autores.

Figura 5 – Modo AM do Processo PORTADORA

```
if(!freq)
{
    freq = 1;
    set_cell(&PORTB, 0);
    set_str(&PORTB, "Mod: AM  P:   Hz");
    //Posicionar-se na 2 lin 1 col
    pula_linha(&PORTB);
    set_str(&PORTB, "Msg:---");
    ADCSRA |= 1 << ADSC;
    valor_ADC = ADC;
    set_cell(&PORTB, 11);
    mapdec(&PORTB,valor_ADC, &freq_P);
}
else
{
    ADCSRA |= 1 << ADSC;
    novo_valor_ADC = ADC;
    if(novo_valor_ADC != valor_ADC)
    {
        valor_ADC = novo_valor_ADC;
        set_cell(&PORTB, 11);
        mapdec(&PORTB,valor_ADC, &freq_P);
    }
}
//Aciona a flag caso seja pressionado algum botao
if(PINC & 1 << 4)
flag_m = 1;
else if(PINC & 1 << 2)
flag_r = 1;
```

Fonte: Autores.

A figura 5 mostra a lógica de funcionamento desse modo na configuração AM. Veja que as funções de escrita no *display* continuam sendo usadas para definir o *layout* no LCD, sendo que após escrever a primeira leitura, há um novo acionamento do ADC, lendo e guardando o valor recebido na variável "novo_valor_ADC". Esse novo acionamento serve para comparar se o primeiro valor lido é igual ao último, para que só seja feita uma nova escrita no LCD caso o valor mude. A intenção dessa implementação é evitar uma atualização desnecessária do *display*, que gere um uso desnecessário de processamento.

1.2.3 Processo RUN

No processo RUN é feita a alteração da porta de leitura do conversor A/D, mudando a leitura do potenciômetro para o sinal modulante, além de dividir esse sinal por 4, para realizar o ajuste de tamanho de 10 bits para 8 bits.

Depois das etapas anteriormente citadas, são definidas todas as equações modulantes, onde cada equação está em uma condição e essa condição é baseada na *flag* do tipo de modulação, que será definida pelo usuário. Por fim, ao começar a realizar a modulação o led RGB altera para a cor azul.

Figura 6 – Definições das Equações Modulantes

```
if(flag_AM)
{
    portadora = cos(2*M_PI*freq_P*t);
    output = (0.5*valor_ADC)*portadora + offset;
}
else if(flag_FM)
{
    output = Ap * cos(2 * M_PI * t * (freq_P + valor_ADC)) + offset;
}
else if(flag_ASK)
{
    portadora = cos(2 * M_PI * freq_P * t);
    output = (0.5 * valor_ADC) * portadora + offset;
}
else if(flag_FSK)
{
    for(unsigned i=0 ; i<8 ; i++){
        if(valor_ADC & 1 << i){
            output =  Ap *cos(4 * M_PI * t * freq_P) + offset;
        }
        else{
            output =  Ap * cos(2 * M_PI * t * freq_P) + offset;
        }
    }
}
```

Fonte: Autores.

1.3 Modulações

Para que as modulações fossem realizadas foi preciso definir o modo de operação do conversor AD do microcontrolador, afinal o sinal de entrada foi inserido no canal 0 do AD.

Assim, no registrador "ADMUX" o bit "REFS0" foi definido como 1 para que a voltagem de referência do conversor fosse a AV_{cc}. Já o registrador "ADCSRA" teve os bits: "ADEN" (*enable* do conversor); "ADCS" (bit responsável por iniciar a conversão); "ADATE" (*enable do auto triggering*); "ADPS2" (para operar com *prescale* 16) todos em nível lógico alto. O registrador "ADCSRB" ficou com os valores *default*.

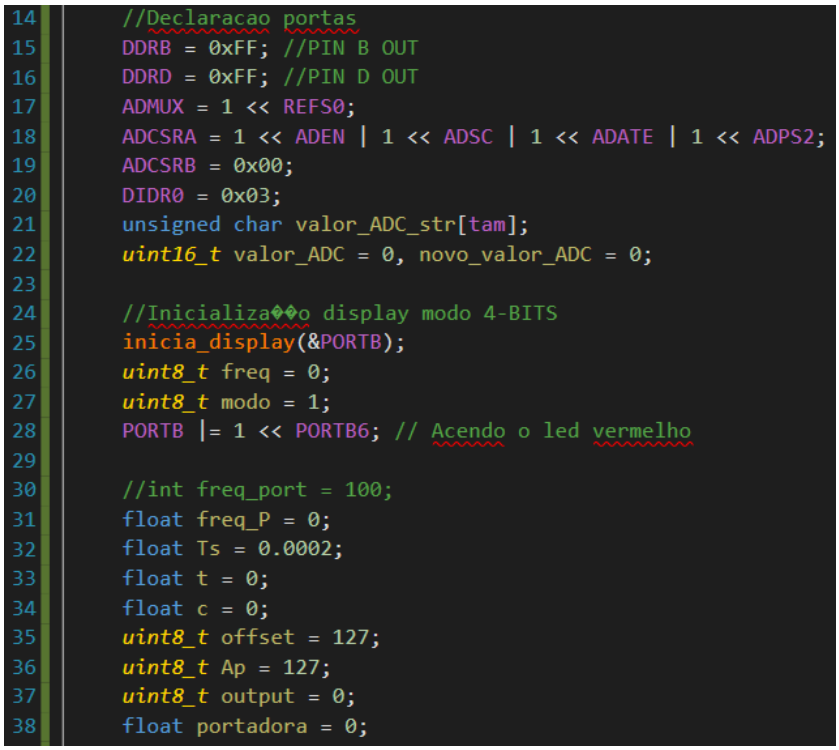
Além disso, foi definido o *offset* como 127 ($\frac{255}{2}$) e as variáveis auxiliares foram inicializadas em 0. Por último, foi definida a taxa de amostragem (T_s) considerando a frequência de Nyquist. Isso determina que a frequência de amostragem utilizada deve ser corresponde a, pelo menos o dobro do valor da frequência máxima da portadora (999). Para melhorar a precisão utilizamos 5 vezes mais, dessa forma, temos o T_s como mostrado

abaixo.

$$T_s = \frac{1}{5 * 999} \tag{1.3}$$

$$T_s = 0.0002 \tag{1.4}$$

Figura 7 – Configurações iniciais



```
14      //Declaracao portas
15      DDRB = 0xFF; //PIN B OUT
16      DDRD = 0xFF; //PIN D OUT
17      ADMUX = 1 << REFS0;
18      ADCSRA = 1 << ADEN | 1 << ADSC | 1 << ADATE | 1 << ADPS2;
19      ADCSRB = 0x00;
20      DIDR0 = 0x03;
21      unsigned char valor_ADC_str[tam];
22      uint16_t valor_ADC = 0, novo_valor_ADC = 0;
23
24      //Inicializa��o display modo 4-BITS
25      inicia_display(&PORTB);
26      uint8_t freq = 0;
27      uint8_t modo = 1;
28      PORTB |= 1 << PORTB6; // Acendo o led vermelho
29
30      //int freq_port = 100;
31      float freq_P = 0;
32      float Ts = 0.0002;
33      float t = 0;
34      float c = 0;
35      uint8_t offset = 127;
36      uint8_t Ap = 127;
37      uint8_t output = 0;
38      float portadora = 0;
```

Fonte: Autores.

1.3.1 Modulação em Amplitude (AM) e por Chaveamento de Amplitude (ASK)

Primeiramente, a entrada do canal 0 do ADC é guardada na variável valor_ADC e é dividida por 4 (para tornar 10 bits em 8bits). Feito isso é utilizada a equação 1.5 para definir a portadora, onde "freq_port" é definida pelo usuário e o "t" irá de 0 a 1 incrementando T_s a cada iteração.

$$portadora = \cos(2\pi * freq\_port * t) \tag{1.5}$$

Na modulação AM a amplitude do sinal de saída é a amplitude da portadora alterada de acordo com o sinal de entrada. Então, o sinal modulado será definido pela

equação 1.6. Note que o valor_ADC é multiplicado por 0.5 para garantir o envelopamento do sinal e que o *offset* é somado para deslocar o sinal em y.

$$output = (0.5 * valor\_ADC) * portadora + offset \tag{1.6}$$

A mesma equação 1.6 foi utilizada para realizar a modulação ASK. Como o sinal de entrada é digital, o sinal modulado é o *offset* quando a entrada estiver em nível lógico baixo e é a portadora quando em nível lógico alto.

1.3.2 Modulação em Frequência (FM)

Na modulação FM a frequência do sinal modulado é a frequência da portadora variando de acordo com o sinal de entrada. Nesse sentido, usamos a equação 1.7, onde A_p é a amplitude da portadora (cujo valor é 127) e, novamente, o *offset* é somado ao sinal de saída.

$$output = A_p * cos(2\pi * t * (freq\_port + valor\_ADC)) + offset \tag{1.7}$$

1.3.3 Modulação por Chaveamento de Frequência (FSK)

A modulação FSK utiliza a mesma portadora da modulação FM, com exceção do valor_ADC que não entra na equação, entretanto, como essa modulação é feita para sinais de entrada digital, quando o bit é 1 a frequência da portadora, que é definida pelo usuário, é multiplicada por dois, como mostra a equação 1.8.

No caso do bit ser zero, a portadora se mantém com a frequência setada pelo usuário, como mostra a equação 1.9

$$output = A_p * cos(4\pi * freq\_port * t) + offset \tag{1.8}$$

$$output = A_p * cos(2\pi * freq\_port * t) + offset \tag{1.9}$$

1.4 Biblioteca do display

Durante a implementação, devido ao extenso código, foi necessário elaborar uma biblioteca que contém as funções com os principais trechos de códigos utilizados, mais especificamente na comunicação entre o ATMega328P e o display. A figura 8 mostra o arquivo ".h" da biblioteca, nela é possível verificar as funções criadas.

Figura 8 – Biblioteca de controle do display

```
#ifndef CONTROLEDISPLAY_H_
#define CONTROLEDISPLAY_H_

#define F_CPU 8000000UL
#define tam 3
#include <avr/io.h>
#include <util/delay.h>
#include <math.h>


void en_up_down(uint8_t *Port);
void inicia_display(uint8_t *Port);
void set_display(uint8_t *Port, unsigned char val);
void pula_linha(uint8_t *Port);
void limpa_lcd(uint8_t *Port);
void set_str(uint8_t *Port, unsigned char *str);
void converte_int_str(uint16_t valor_ADC, unsigned char *str);
void set_cell(uint8_t *Port,  unsigned int val);
void val_binario(uint8_t *Port, uint8_t val);
void val_decimal(uint8_t *Port, uint8_t val);
void limpa_cursor (uint8_t *Port);
void mapdec(uint8_t *Port, uint16_t valor_ADC, float *freq_P);
void set_freq(uint8_t *Port, unsigned char *str);
#endif /* CONTROLEDISPLAY_H_ */
```

Fonte: Autores.

As funções, no geral, possuem como um de seus parâmetros um ponteiro para uma variável do tipo *uint8_t*, que por sua vez representa as portas do *PORTB*, já que a comunicação com o display é feita atráves dele.

As funções "en_up_down", "inicia_display", "pula_linha", "limpa_lcd" e "limpa_cursor", não recebem parâmetros extras, visto que seus códigos internos se resumem a enviar comandos específicos para o display, sem que seja feito qualquer tipo de cálculo prévio.

A função "set_display" recebe um caractere que deve ser escrito no visor do display e, por meio de comandos de seleção, escolhe o valor adequado para representar tal elemento na tela. De forma semelhante, "set_cell" define o valor que deve ser transmitido ao display, baseado na posição que se deseja acessar, passada como parâmetro da função.

No envio de números, é feito um tratamento prévio, antes do envio ao display, onde os valores numéricos são convertidos em uma *string*, este processo é realizado em "converte_int_str".

As funções "set_str" e "set_freq" possuem códigos similares, a diferença é que a primeira é voltada para a introdução de *strings* no geral, em que o tamanho do texto pode ser variado, enquanto a segunda função é utilizada durante a escolha da frequência da portadora, onde apenas três dígitos, obrigatoriamente, serão escritos no display.

Para representar o valor do sinal de saída, foram elaboradas as funções "val_binario"

e "val_decimal", que são utilizadas nas modulações digitais e analógicas, respectivamente.

Por fim, foi imposto que durante a escolha da frequência da portadora, os valores poderiam variar entre 100 Hz e 999 Hz, nesta implementação a função responsável por este tratamento é a "mapdec". Esta função recebe o valor convertido pelo ADC e o mapeia no intervalo de valores possíveis, utilizando de algumas das funções já citadas para o envio da informação na sequência.

1.5 Saturação do Potenciômetro

A saturação do potenciômetro é necessária para que o potenciômetro, que atua entre 0 - 1023, atue no intervalo de 100 a 999, conforme o projeto exige. Para isso, foi feita uma função que tem como parâmetros: o valor de entrada do potenciômetro (valor_ADC); um ponteiro que aponta para a porta que seta os bits de entrada do LCD (*port) e um ponteiro que aponta para frequência da portadora (*freq_P).

Com o valor_ADC, é feita a conversão do valor de entrada para o intervalo de 000 a 999, como mostra a equação 1.10.

$$aux\_adc = \frac{(float)valor\_ADC}{1023} * 899 + 100 \tag{1.10}$$

No endereço "freq_P" é guardado o valor calculado (aux_adc) e este é convertido para um valor em decimal de acordo com a tabela ASCII, para tornar possível a exibição no *display*. Feito isso, o valor convertido é setado na porta de saída para o LCD 16×2, viabilizando a exibição correta dos valores.

Figura 9 – Mapeamento do potenciômetro

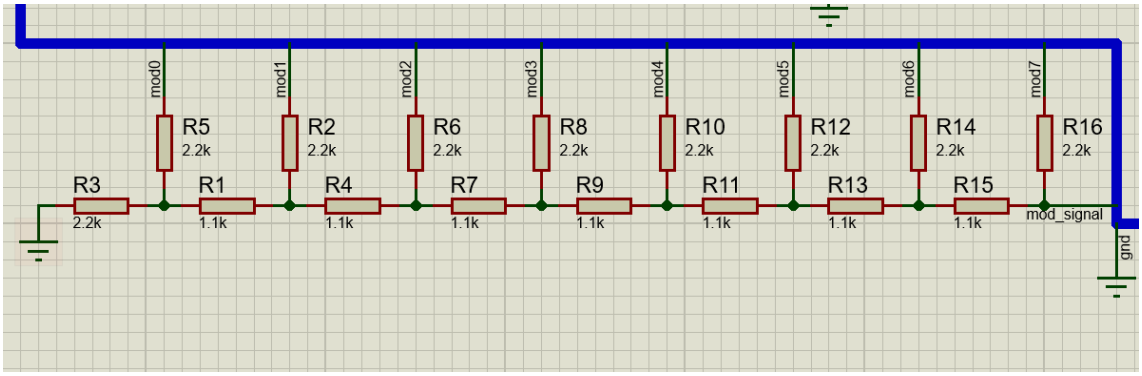
```
347 void mapdec(uint8_t *Port, uint16_t valor_ADC, float *freq_P)
348 {
349     unsigned char aux_saida[tam];
350     unsigned char saida[tam];
351     float aux_adc=0;
352
353     aux_adc = ((((float)valor_ADC/ 1023)* 899) + 100);
354     *freq_P = aux_adc;
355     converte_int_str(*freq_P, &saida);
356     set_freq(Port, saida);
357 }
```

Fonte: Autores.

1.6 Rede R2R

A rede R2R foi montada conforme projetado, para isso foi utilizado resistores de 1.1kΩ e de 2.2kΩ. A montagem é a mostrada na Figura 10.

Figura 10 – Rede R2R



Fonte: Autores.

2 RESULTADOS

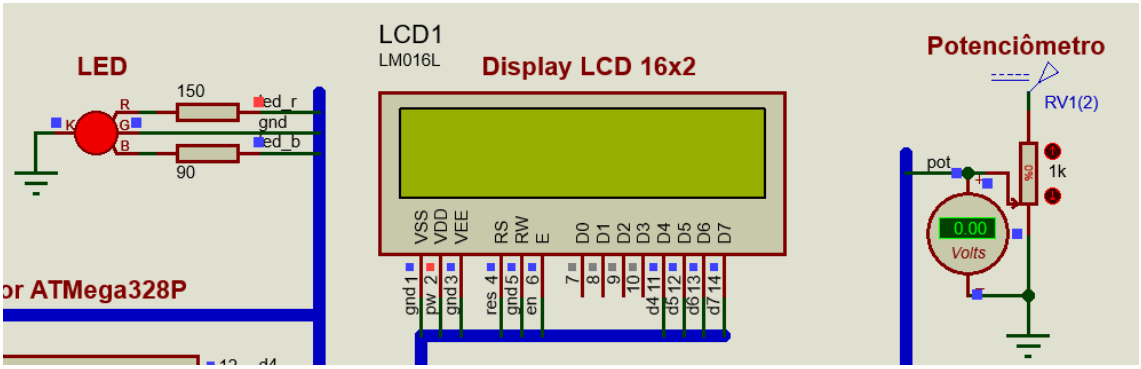
Uma vez implementado o código na linguagem C, foi gerado um programa no formato .HEX para realizar sua simulação no ATMega328p com todos os periféricos conectados, no *software* Proteus.

2.1 Simulação no Proteus

Todo o esquemático do projeto foi desenvolvido no Proteus, em sua versão 8.9. O esquemático final está disponível no Anexo B.

Ao se iniciar a simulação, o display LCD é ligado, mas permanece sem nenhuma informação mostrada nele e o led vermelho é aceso, indicando que nenhuma modulação está ocorrendo, como mostra a Figura 11.

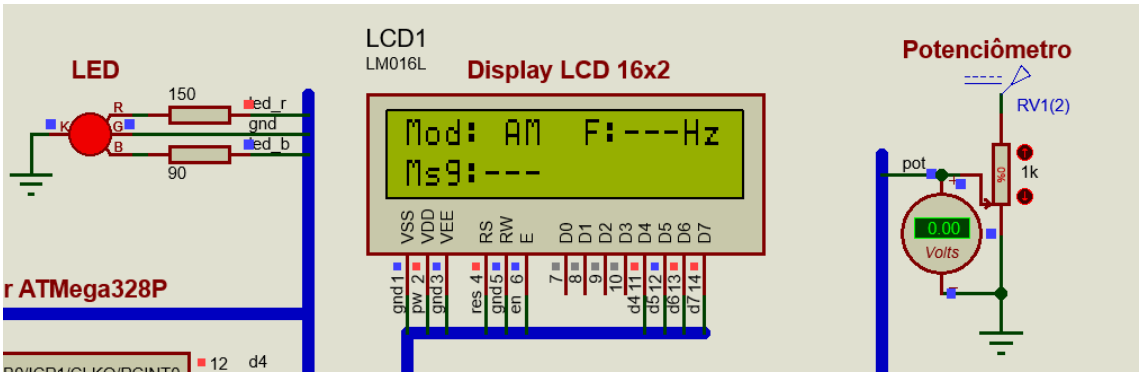
Figura 11 – Simulação do estado Inicial



Fonte: Autores.

Após o usuário clicar no botão de modulação, o display fornece as informações da modulação que está sendo selecionada, podendo o usuário alternar entre os quatro tipos possíveis mudando o valor de resistência do potênciometro. Neste estado, apesar do usuário já poder selecionar o tipo de modulação a ser executada pelo SDR, o led ainda continua na cor vermelha, indicando que ainda não se iniciou a modulação, como mostra a Figura 12.

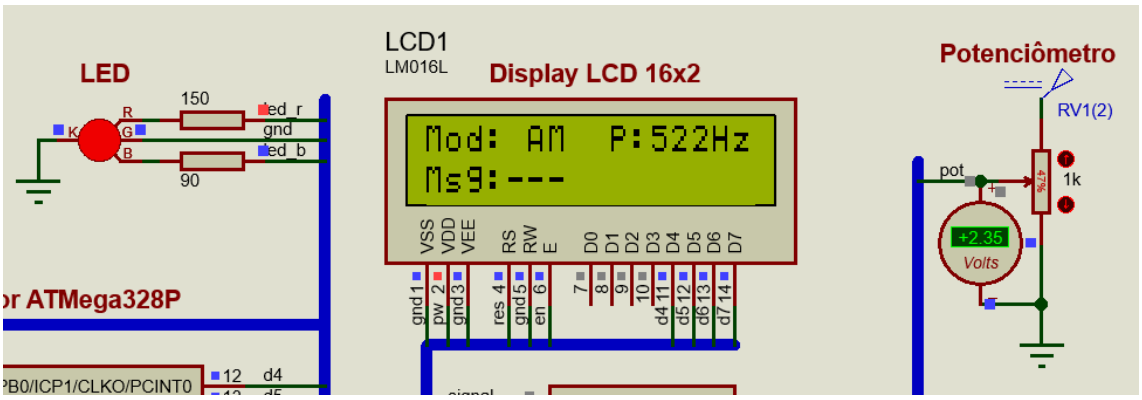
Figura 12 – Simulação do estado Modulação



Fonte: Autores.

Com o tipo de modulação selecionada, o usuário precisa pressionar o botão P, responsável por ir para o estado de seleção da frequência da portadora, neste estado o display fornece ao usuário a possibilidade de escolher entre um número de 100Hz∼999Hz para a frequência da portadora. O periférico de conversor A/D do ATMega328P é capaz de converter tensões de 0V a 5V, sendo 5V igual a 1023 em binário na saída do conversor, logo, para que seja saturado o intervalo da problemática foi utilizado uma função previamente discutida neste relatório. A Figura 13 demonstra as informações disponíveis no display nesse estado, repare que o led ainda continua aceso na cor vermelha.

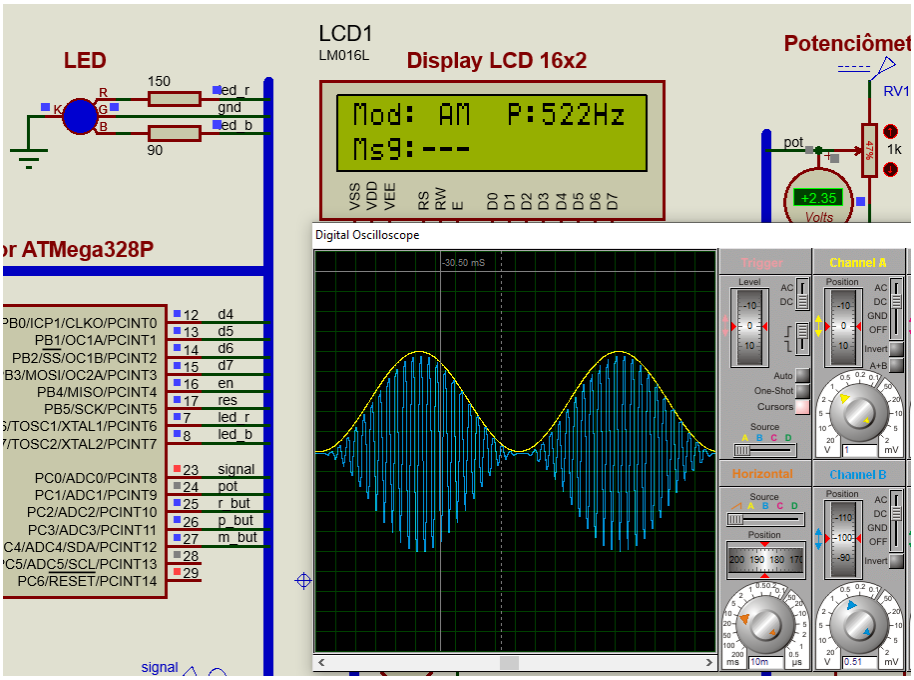
Figura 13 – Simulação do estado Portadora.



Fonte: Autores.

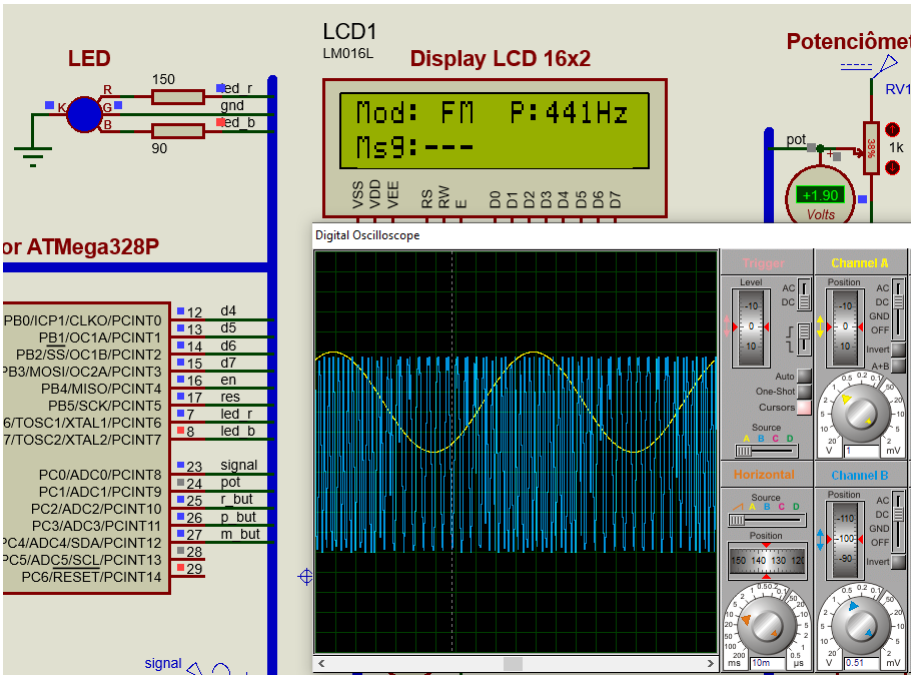
Por fim, ao usuário pressionar o botão R depois de selecionar entre os quatro tipos de modulações disponíveis e a frequência da portadora, o SDR vai para o estado de RUN, onde a modulação é iniciada e o led azul é aceso, como mostra a Figura 14, 15, 16 e 17.

Figura 14 – Simulação da Modulação AM



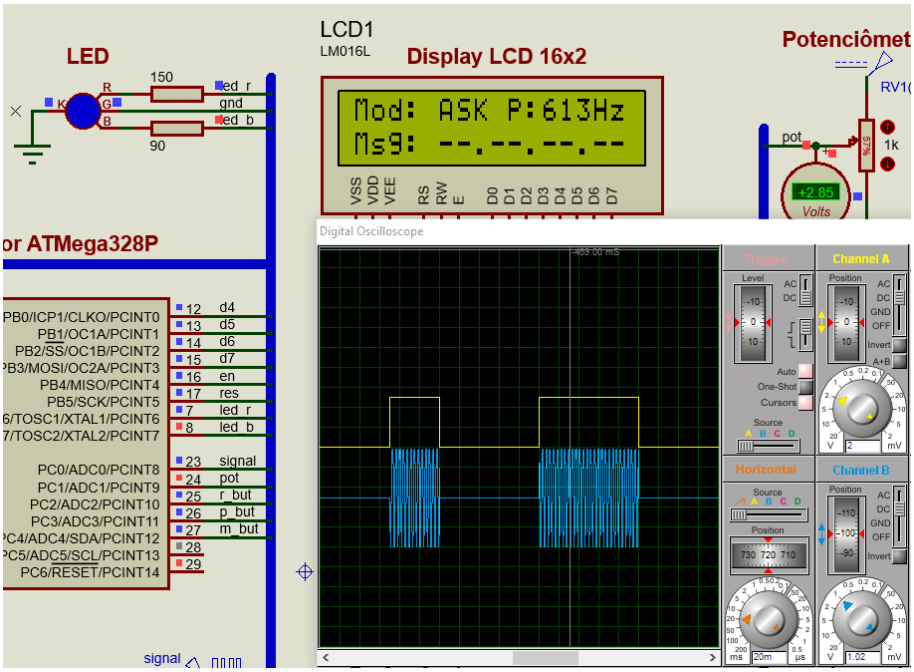
Fonte: Autores.

Figura 15 – Simulação da Modulação FM



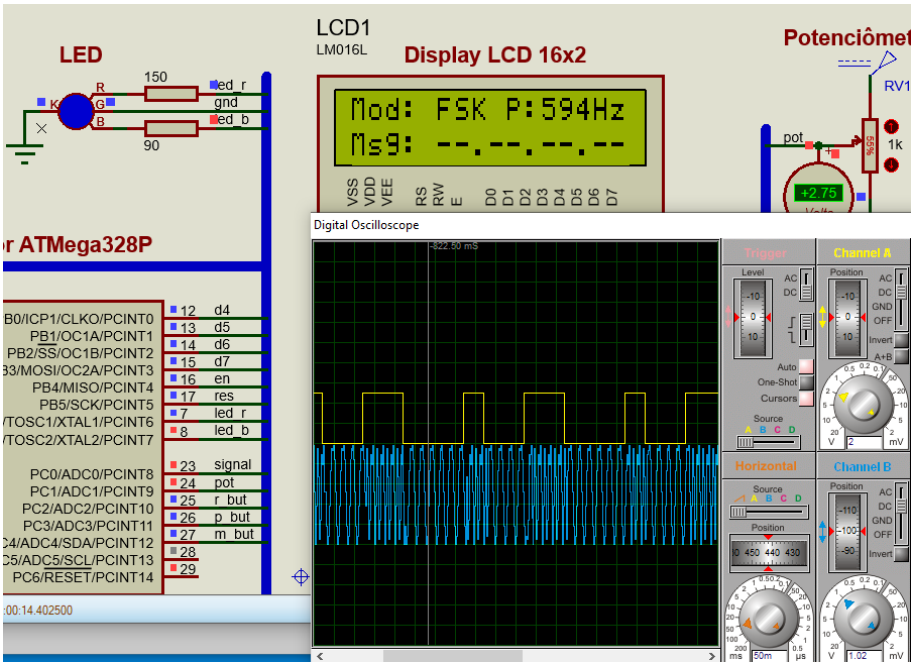
Fonte: Autores.

Figura 16 – Simulação da Modulação FSK



Fonte: Autores.

Figura 17 – Simulação da Modulação FSK



Fonte: Autores.

[IMAGEM DA MODULAÇÃO SENDO EXECUTADA AQUI]

3 CONCLUSÃO

Através da implementação, foram adquiridos conhecimentos e habilidades acerca de programação em microcontroladores ATMega 328p, na linguagem C. O projeto ainda permitiu que fosse compreendido o funcionamento dos periféricos utilizados, tais como o circuito R2R, e o módulo LCD. O módulo foi operado durante, praticamente, todos os processos existentes no sistema implementado, onde foi possível manipular o seu funcionamento de acordo com as necessidades encontradas. Com os aprendizados obtidos durante o entendimento do projeto, ficou mais clara a percepção das modulações realizadas durante as simulações, das quais, foi possível aferir os resultados obtidos de acordo com o que era esperado.

ANEXO A – Relato semanal

Líder: Isaac de Lyra Junior

A.1 Equipe

Tabela 1 – Identificação da equipe

Função no grupo	**Nome completo do aluno**
Redator	Albertho Síziney Costa
Debatedor	Lucas Batista da Fonseca
Videomaker	Sthefania Fernandes Silva
Auxiliar	Lucas Augusto Maciel da Silva

Fonte: Produzido pelos autores.

A.2 Defina o problema

O problema consistiu na implementação de um mini rádio definido por software (SDR - *Software Defned Radio*). Para a realização da tarefa foi utilizado um circuito baseado em um uC AVR (ATMega328P), o SDR deveria ser capaz de enviar informações analógicas ou digitais dos objetos, sendo moduladas em quatro técnicas diferentes de modulação (AM, FM, ASK, FSK).

Para cada uma das técnicas de modulação, o SDR apresentaria em um display LCD 16x2 o valor da mensagem a ser enviada em binário, no caso das analógicas, seria um valor inteiro de 8 bits correspondente a amplitude de uma amostra da mensagem a ser enviada, e nos casos digitais o conjunto de 8 bits da mensagem a ser enviada.

O usuário poderia escolher entre os quatro tipos de modulações possíveis, como também a frequência da portadora, que poderia variar entre 100Hz∼999Hz. A mensagem a ser enviada entra no canal 0 do periférico de conversão A/D do microcontrolador, e como saída o SDR utiliza um display LCD 16x2 e uma saída analógica de 8 bits através de um conversor D/A implementado por uma rede R2R.

Todo o código fonte utilizado para a implementação foi escrito na linguagem C.

A.3 Registro de *brainstorming*

A primeira reunião foi marcada para a quinta-feira (12/08/2021) para que desse tempo de todos estudarem o projeto recebido e se familiarizar com a linguagem C novamente.

Nesta reunião foi iniciado a discussão acerca do display LCD 16x2 e os próximos passos para a implementação, foi decidido que iríamos focar na criação da biblioteca responsável pelo display LCD até a próxima reunião.

Na reunião seguinte, realizada na sexta-feira (13/08/2021) a biblioteca responsável por enviar informações para o display estava quase concluída, restando apenas alguns bugs a serem corrigidos posteriormente. Nesta reunião foi decidido que teríamos duas equipes, uma seria responsável por resolver os bugs restantes da biblioteca do display e a outra seria responsável por desenvolver o código que faria as modulações pedidas no problema.

As reuniões do sábado (14/08/2021) e domingo (15/08/2021) serviu apenas para continuar o trabalho das reuniões anteriores, no domingo houve a identificação de que algumas das fórmulas de modulações passadas no projeto não estavam funcionando na implementação, logo foram modificadas e conseguimos realizar todas as modulações.

A reunião da segunda (16/08/2021) serviu para reunir as duas equipes e apresentar os resultados dos trabalhos. Esta reunião também serviu para escrever o relatório de implementação.

A.4 Pontos-chaves

Um dos maiores pontos-chaves foi a compreensão da linguagem C e como ela deve ser utilizada para microcontroladores. Também foi bastante importante os estudos acerca de como o display LCD 16x2 funciona e os comandos necessários para enviar informações para o mesmo. Por fim, o conhecimento de modulações foi de suma importância para a realização da implementação deste problema.

A.5 Questões de pesquisa

Para que a implementação fosse bem sucedida, os seguintes temas foram estudados:

- Linguagem C para Microcontroladores
- Criação de Bibliotecas na linguagem C
- Modulação em Frequência (FM, FSK)
- Modulação em amplitude (AM, ASK)
- Display LCD 16x2
- Rede R2R (Conversão D/A)
- Conversão A/D

A.6 Planejamento da pesquisa

Primeiro foi estudado sobre a utilização da linguagem C para microcontroladores e como se pode criar bibliotecas para evitar retrabalho e organizar o código. Em seguida, foi estudado como o display LCD 16x2 funciona e os comandos necessários para que ele possa funcionar da forma que o problema pede. Por fim, foi estudado sobre as conversões A/D e D/A, bem como as modulações em amplitude (AM, ASK) e frequência (FM, FSK). Para todos os estudos foi utilizado vídeos do Youtube, fóruns sobre microcontroladores e livros/artigos sobre os temas abordados no problema.

ANEXO B – Esquemático