GABRIEL PINHEIRO CAVALCANTI GUERRA SEABRA
IAGO PEREIRA CASSEL
JACIMARA VITORIA NUNES PACHECO
JARDEL GREGÓRIO DO NASCIMENTO

GRUPO 1: IMPLEMENTAÇÃO DO PROJETO DE UM MINI SDR

Brasil

novembro de 2020

GABRIEL PINHEIRO CAVALCANTI GUERRA SEABRA
IAGO PEREIRA CASSEL
JACIMARA VITORIA NUNES PACHECO
JARDEL GREGÓRIO DO NASCIMENTO

GRUPO 1: IMPLEMENTAÇÃO DO PROJETO DE UM MINI SDR

Décimo relatório apresentado à disciplina de Sistemas Digitais, correspondente ao semestre 2020.6 do curso de Engenharia Mecatrônica da UFRN, referente a implementação do projeto de um SDR (Software Defined Radio)

Universidade Federal do Rio Grande do Norte – UFRN

Curso de Engenharia Mecatrônica

Disciplina de Sistemas Digitais

Brasil

novembro de 2020

Resumo

O seguinte trabalho trata-se da implementação do projeto de um sistema de SDR utilizando o Atmega328P, em que é realizado a demodulação de sinais modulados por amplitude, analógicos ou digitais.

Palavras-chaves: Atmega328P, demodulação.

Lista de ilustrações

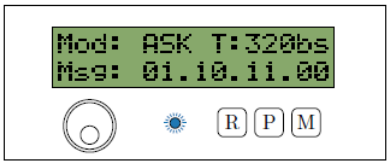
Lista de tabelas

Sumário

1 Introdução

O projeto apresentado a seguir é uma implementação de um mini SDR, que basicamente é um sistema de radiocomunicação, o qual realiza funções por software, normalmente executadas em Hardware. O objetivo principal do sistema implementado, é realizar demodulação de sinais modulados em AM e ASK.

Figura 1 – Interface homem/máquina do circuito



Fonte: Disponibilizado pelo professor

No sistema deve-se pressionar o botão **M**, para definir o tipo de modulação AM ou FM, por meio do potenciômetro. Após isso, pressiona o botão **P**, para definir a frequência da portadora, também pelo potenciômetro, que pode ser de 5KHz à 50KHz. Essas configurações são apresentadas no display LCD.

Obtendo todas essas informações é realizada a demodulação do sinal e o resultado é mostrado no display LCD. Adiante será apresentado o funcionamento desse sistema em relação ao código, hardware e resultados de simulações.

2 Metodologia

A partir das definições do projeto recebido da semana 9, foi realizado a implementação utilizando as ferramentas **Proteus e Matlab**, para a simulação e o **Atmel Studio 7.0** para verificação do código em C.

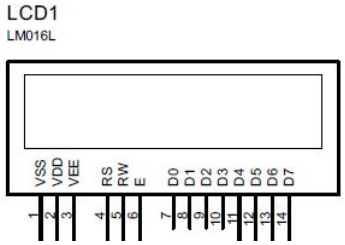
Foram analizados todos os pontos, para saber o que poderia ser implementado e os que deveriamos fazer modificações para que a implementação se tornasse possível. Foram encontrados três mudanças: modificação da MDE para lógica correta; redução de pinos para controle do led; correção do clock.

3 Componentes

3.1 Display LCD 16x2

O LCD (Liquid Crystal Display), é um dispositivo eletrônico usado para exibição de dados. Esse tipo de Display apresenta dezesseis colunas e duas linha, isso significa que podem ser apresentados nele, 32 caracteres.

Figura 2 – Display LCD 16x2



Fonte: Elaborado pelo autor

3.1.1 Tabela de pinagem do Display LCD 16x

Tabela 1 – Tabela da pinagem utilizada no ATmega328P

Pinos	I/O	Descrição
VSS	–	Terra
VDD	–	Alimentação
VEE	–	Controle para contraste da tela
RS	In	Registrador Seletor
RW	In	Ler/Escrever
E	In/Out	Habilitador
D0	In/Out	Dado de 8 bits LSB
D1	In/Out	Dado de 8 bits
D2	In/Out	Dado de 8 bits
D3	In/Out	Dado de 8 bits
D4	In/Out	Dado de 8 bits
D5	In/Out	Dado de 8 bits
D6	In/Out	Dado de 8 bits
D7	In/Out	Dado de 8 bits MSB

Fonte: Elaborado pelo autor

Nesse projeto, utilizou-se a entrada *RW* em modo baixo, pois só será realizado escrita no LCD. A entrada *E* é usado para travar as informações apresentadas aos pinos de dados. Quando os dados são enviados para os pinos, um pulso de HIGH para LOW deve ser aplicado à este pino para que retenha os dados presente nos pinos de dados.

Para escrever ou ler dados, é usado as portas do D0 ao D7, porém como no Atmega328P temos uma quantidade limitada de portas, optou-se escrever os dados no LCD utilizando pelo método de transmissão de dados de 4 bits.

3.1.2 Código de comandos do LCD

Tabela 2 – Código de comandos do LCD

Códigos(HEX)	Comando para o LCD
01	Limpa a tela do display
02	Retorna para Home
04	Decremento do cursor(muda cursor para esquerda)
06	Incremento do cursor(muda cursor para direita)
05	Muda display para direita
07	Muda display para esquerda
08	Display off/Cursor off
0A	Display off/Cursor on
0C	Display on/Cursor off
0E	Display on/Cursor piscando
0F	Display on/Cursor piscando
10	Muda a posição do cursor para esquerda
14	Muda a posição do cursor para direita
18	Muda todo o display para a esquerda
1C	Muda todo o display para a direita
80	Força o cursor para o início(1^{a} linha)
60	Força o cursor para o início(2^{a} linha)
28	2 linhas e matriz 5x7(D4-D7, 4 bit)
38	2 linhas e matriz 5x7(D0-D7, 8 bit)

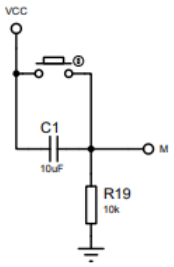
Fonte: Elaborado pelo autor

No nosso modo de transmissão devemos levar o pino RS para o nível lógico 0 para configurar o LCD ou 1 para escrever. Em seguida transfere-se para os pinos D[4:7] os 4 bits mais significativos do dado (nibble maior), em seguida um pulso de curta duração no enable. Em seguida é repassado o lower nibble, os 4 bits menos significativos. Outro pulso no enable é dado. Com esse processo em loop podemos escrever repetidamente no LCD.

3.2 *PushButtons*

Para os botões utilizados, optou-se utilizar a configuração *pull-down*, em que quando não estão pressionandos, suas saídas ficam em baixo, caso contrário, a saída será em alto. A figura abaixo, apresenta o circuito para estes dispositivos.

Figura 3 – Configuração do pushButton em pull-down

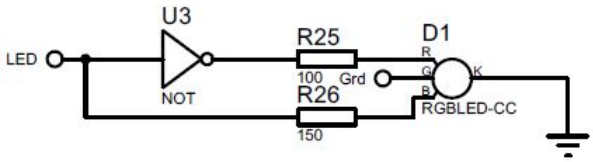


Fonte: Elaborado pelo autor

3.3 LED RGB

O LED utilizado neste sistema é do tipo RGB. As cores utilizadas no sistema são vermelho e azul, como estas são cores primárias do sistema RGB, basta apenas uma alimentação simples para representa-las, a Figura a seguir apresenta a configuração do circuito do LED utilizado no projeto.

Figura 4 – Circuito do LED



Fonte: Elaborado pelo autor

Utilizou-se apenas uma porta do Atmega328P, para a determinar as cores a serem sinalizadas. Quando em nível alto será alimentada a porta azul, quando em baixo a porta vermelha. A cor verde nunca será utilizada, por isso é aterrada.

3.4 Potenciômetro

Um potenciômetro será usado para ajuste do valor da portadora e para definir o tipo de modulação. No ajuste da portadora o valor de 0V representa 5kHz e 5V, 50KHz.

No potênciômetro usado no proteus, pode-se obete 100 valores diferentes, a cada 1% é acrecentado 0.05V e o valor so ADC vai de 0 até 1024.

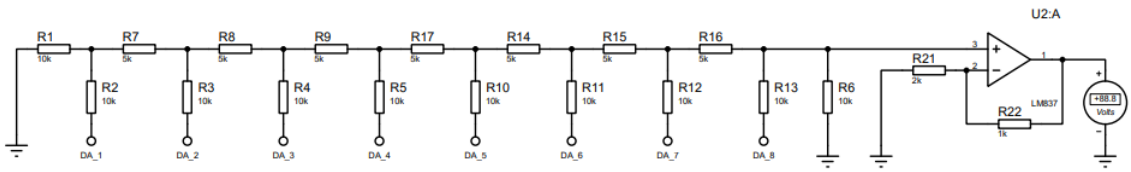
Figura 5 – Potenciômetro

Fonte: https://www.baudaeletronica.com.br/potenciometro-linear-de-10k-10000.html

3.5 Conversor Digital-Analógico com a rede R/2R

A malha R/2R converte um sinal digital, para analógico. Este circuito é uma associação entre resistores, por isso é de fácil implementação. Dessa forma, como os componentes eletrônicos não entendem sinais digitais, eles representam os sinais de forma analógica.

Figura 6 – Circuito do R/2R



Fonte: Elaborado pelo autor

A rede R/2R no projeto, terá a função de gerar um sinal analógico, a partir do sinal de 8 bits que foi convertido para digital, após a demodulação.

4 Desenvolvimento

4.1 Máquina de estados (MDE)

No estado **START** é configurado o conversor dos valores do potenciômetro, inicia as entradas analógicas e digitais, inicia o Display LCD e é colocado um protótipo de mensagem AM, logo em seguida vai para o estado **WAIT**.

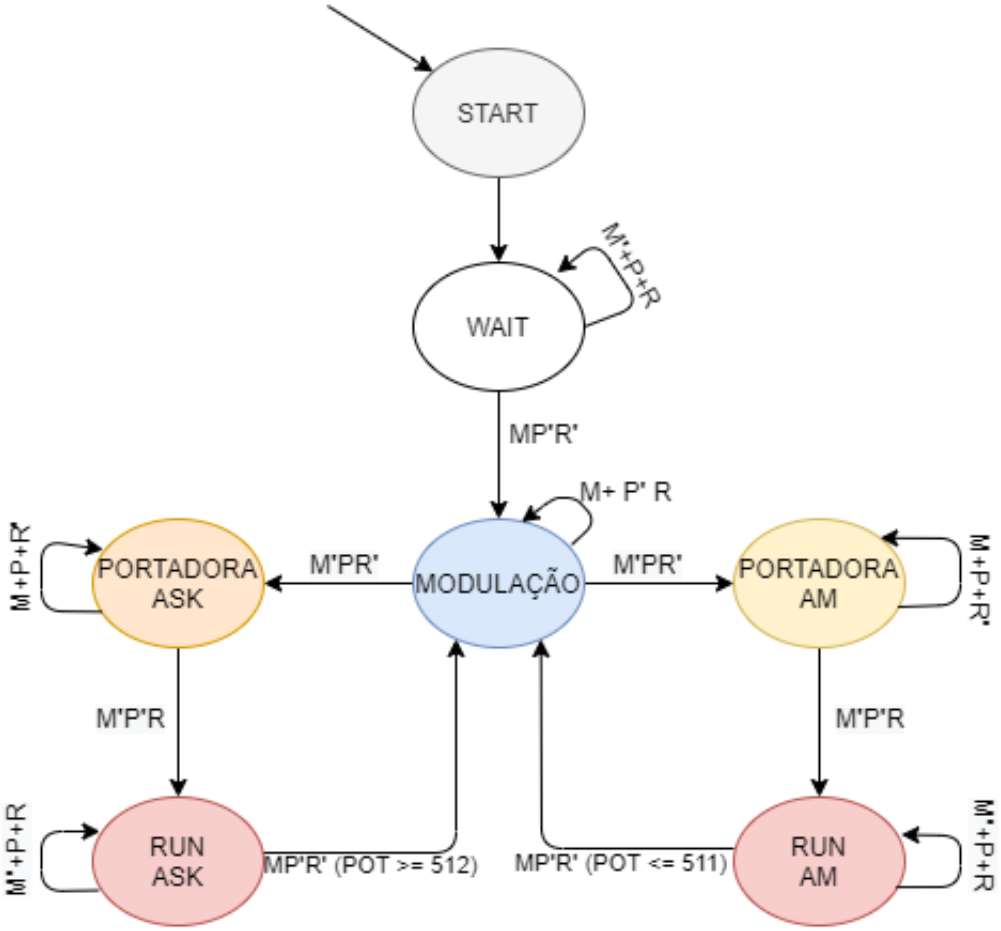
No estado de **WAIT**, espera-se até o botão **M** ser pressionado, para o sitema ser direcionado para o estado de **MODULAÇÃO**.

No estado de **MODULAÇÃO** faz-se a leitura do potenciômetro e realiza a conversão AD, durante esse período ele compara, se o resultado for menor ou igual a 511, apresenta no Display a mensagem AM, se maior ou igual à 512, apresenta uma mensagem ASK. Isso ocorre repetidas vezes, até pressionar-se o botão P, o qual direcionará o sistema para o estado da **PORTADORA**, de acordo com o tipo de modulação que foi escolhido.

Em **PORTADORA** é armazenado qual modulação foi escolhida e realiza a leitura do potenciômetro relacionado à qual estado da portadora do sistema está. O valor dessa leitura é introduzido em uma equação, que determina quanto equivale em Hertz. Isso se repete até o botão **R** ser pressionado, caso isso ocorra, será direcionado para o estado **RUN**.

No estado **RUN** é realizado a atualização do filtro e da cor do LED, pressionando o botão **M**, volta para **DEMODULAÇÃO**.

Figura 7 – Máquina de estados



Fonte: Autoria própria

4.2 Clock

Após a discussão feita na semana anterior e a avaliação dos recursos que seriam necessários para realização das conversões analógicas-digitais, decidimos repensar sobre a frequência de clock utilizada. Dentre os fatores que levamos em conta estava a necessidade do ATMega ser capaz de amostrar o sinal da mensagem em uma taxa maior do que a da portadora do sinal (de até 50kHz), levando em conta que a amostragem em uma taxa maior replica o sinal do espectro de frequência, em frequências baixas. Além disso, outro aspecto levado em consideração é o fato da amostragem feita pelo ADC necessitar de 25 ciclos de clock para uma conversão inicial e 13 ciclos para conversões subsequentes, dessa forma o projeto deve ser escalonado para evitar problemas com esse requerimento do conversor. Por fim, além da amostragem, um conjunto de operações devem ser realizadas para demodulação da mensagem, gerando atrasos de pulso de clock em cada interação. Tendo em vista todas essas peculiaridades, foi decidido o uso de uma frequência de clock de 8MHz. A frequência escolhida é relativamente alta, porém garante um range de trabalho seguro para realizar a tarefa.

4.3 Conversão Analógica para digital

A conversão Analógica para digital é necessária para os dados recebidos do potenciômetro assim como do sinal a ser demodulado. O conversor utilizado é do próprio Atmega328P, este apresenta resultado de 10 bits.

No microcontrolador o bit ADLAR do registrador ADMUX, irá ajustar se os bits serão distribuidos nos resistradores ADCH e ADCL à esquerda (ADCH com os 8 MSBs e os bits 7:6 do ADCL com os LSBs), ou não.

Para realizar uma conversão aciona-se o bit ADSC do registrador ADCSRA, este se mantém em 1 até o término da conversão e ao final, automaticamente ele volta para zero. O valor para uma conversão é dado por:

$$ADC = \left(\frac{V_{IN}1024}{V_{REF}}\right)$$

Onde V_{IN} é a tensão para conversão na entrada do pino e V_{REF} é a tensão selecionada de referência. O valor 0x000 representa a tensão zero e o valor 0x3FF representa VREF menos 1 LSB.

4.3.1 Registradores do ADC

No registrador **ADMUX** os **bits 7:6** selecionam a fonte de tensão para o ADC, a tabela a seguir mostra os possiveis valores.

Figura 8 – Seleção da fonte do ADC

REFS1	REFS0	Seleção da Tensão de Referência
0	0	AREF, tensão interna V_{REF} desligada.
0	1	AVCC. Deve-se empregar um capacitor de 100 nF entre o pino AREF e o GND.
1	0	Reservado.
1	1	Tensão interna de referência de 1,1 V. Deve-se empregar um capacitor de 100 nF entre o pino AREF e o GND.

Fonte: Borges de Lima, C; Villaça, M. V. M. AVR e arduino técnicas de projeto.

O b**it 5** (ADLAR), afeta a representação do resultado da conversão dos registradores de dados do ADC. Quando em 1, alinha à esquerda, caso contrário, alinha à direita.

Em relação, aos **bits 3 ao 0** (MUX3:0), esses selecionam qual entrada analógica será conectada ao ADC.

Figura 9 – Seleção do canal de entrada

MUX3..0	Entrada
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	Sensor interno de temperatura
1001-1101	reservado
1110	1,1 V (tensão fixa para referência)
1111	0 V (GND)

Fonte: Borges de Lima, C; Villaça, M. V. M. AVR e arduino técnicas de projeto.

O outro registrador que controla o ADC é o **ADCSRA**. Nesse registrador, o **bit 7** (ADEN) habilita uma conversão, o **bit 6** (ADSC), inicia uma conversão e retorna a zero automaticamente, depois do processo finalizado.

O **bit 5** (ADATE), ativa o modo de auto disparo. O ADC começará uma conversão na borda positiva do sinal selecionado de disparo.

O **bit 4** (ADIF) é ativo quando uma conversão for completada e o registrador de dados atualizado. O **bit 3** (ADIE) habilita a interrupção do ADC após a conversão se o bit I do SREG estiver habilitado. Os **bit 2, 1 e 0** (ADPS2:0) Determinam a divisão do clock da CPU para o clock do ADC.

Figura 10 – Seleção da divisão de clock para o ADC

ADPS2	ADPS1	ADPS0	Fator de Divisão
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Fonte:

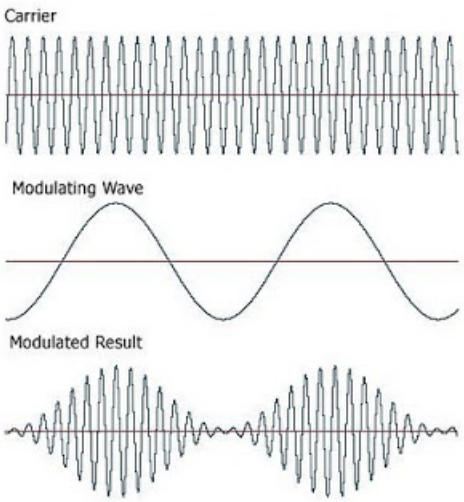
4.4 Demodulação dos sinais

Para demodulação do sinais, utilizou-se sinais modulados por amplitude analógico (AM) e digital (ASK).

4.4.1 Modulação AM e ASK

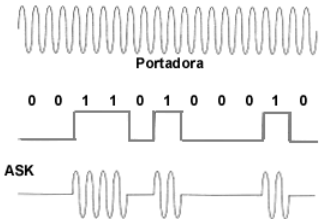
Na modulação AM e ASK, a amplitude da portadora é modificada de acordo com o sinal a ser transmitido, dessa forma, não tem variação na fase, nem na frequência desta. Abaixo as Figuras representam essas modulações.

Figura 11 – Modulação AM



Fonte: https://wiki.sj.ifsc.edu.br/index.php/Modula%C3%A7%C3%A3o_AM_-_t4%C3%A9cnico

Figura 12 – Modulação ASK



Fonte: http://www.dainf.cefetpr.br/ pelisson/redes/modulac.htm

No projeto, para obeter o sinal a ser trabalhado, realiza-se a multiplicação do sinal à ser enviado, pela a portadora, como pode ser visto na equação abaixo para modulação AM.

$$SinalEnviado = (2.5 + 2.5cos(f\_msg * t)) * cos(f\_portadora * t)$$

4.4.2 Escolha do filtro

Foram analisadas duas opções para implementar o filtro passa-baixa. Inicialmente pensamos em implementar um filtro de resposta de impulso (FIR), porém foi identificado que muitas operações de multiplicação e divisão, assim como operações com pontos flutuantes eram necessárias para sua implementção.

Tendo isto, decidimos utilizar um filtro de média movel. Neste, a partir da média de uma série de amostragens, podemos retirar os sinais de alta frequência, restando apenas as tensões de baixa frequência.

Para este projeto, soma-se 16 valores do sinal, divide por 16 e assim obtêm-se a média. Dessa forma, frequências baixas são mantidas, e as altas, descartadas.

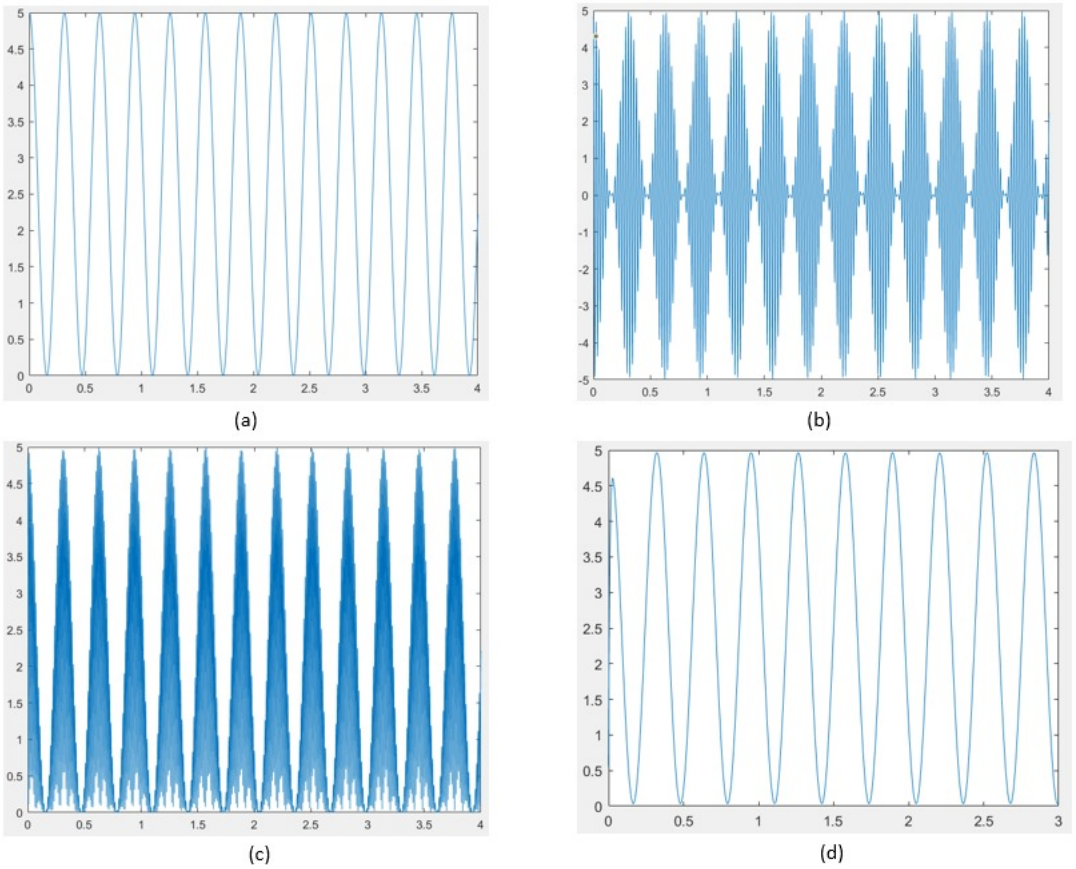
Para ilustração do conceito, podemos exemplificar como é feita a filtragem de um valor n-esimo amostrado utilizando um média móvel com uma janela de 5 unidades. Isso nos informa que o valor n será a soma dos últimos cinco valores amostrados dividido por 5, como visto abaixo.

$$y(n) = (x(n) + x(n-1) + x(n-2) + x(n-3) + x(n-4))/5$$

4.4.3 Utilizando o Matlab para obter demodulação do sinal

Antes da implementação em C, foi implementado no Matlab um código teste, para verificar se o algoritmo da demoludação realmente funcionava assim como o resultado esperado para simulação no AVR.

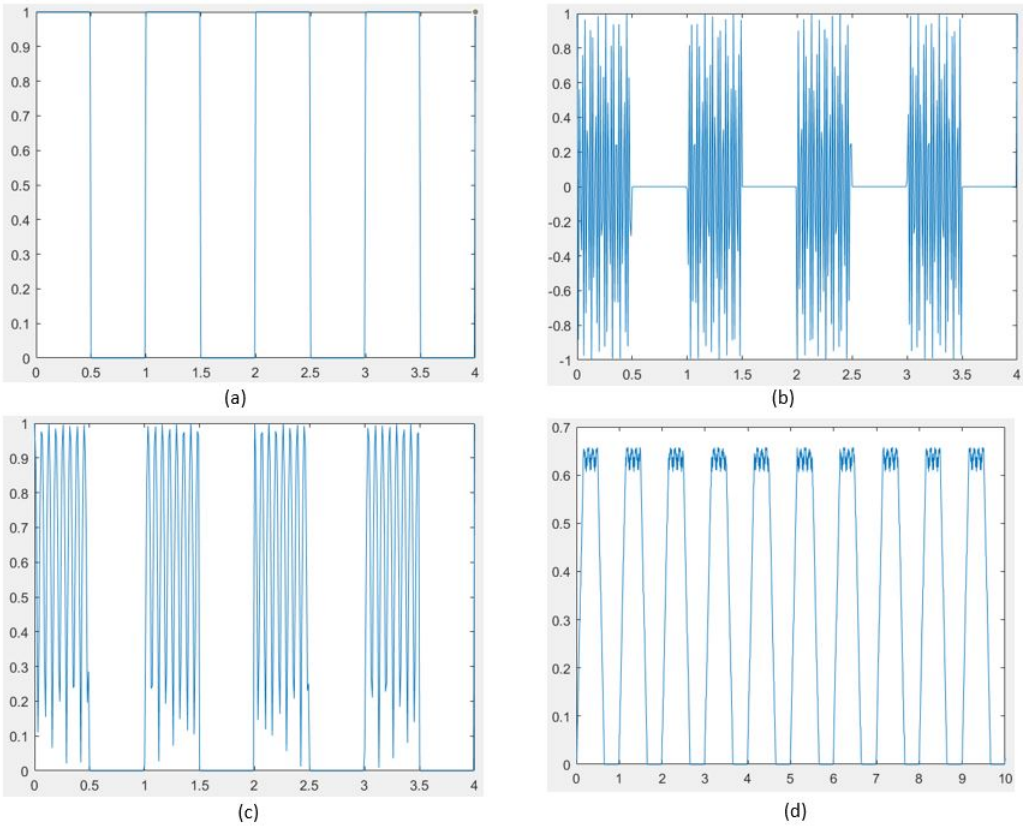
Figura 13 – demodulação AM



Fonte: Autoria própria

Na Figura 13 podemos ver como a demodulação AM utilizando um filtro de média móvel se comporta. Na Figura 12.a *(a)* podemos ver o sinal senoidal original, **(b)** o sinal modulado, **(c)** o modulado retificado e na **(d)** o sinal demodulado. Analisando os gráficos, pode-se perceber que em comparação com o sinal original, o demodulado é idêntico, contendo apenas um pequeno atraso e uma leve atenuação nos primeiros sinais.

Figura 14 – Demodulação ASK



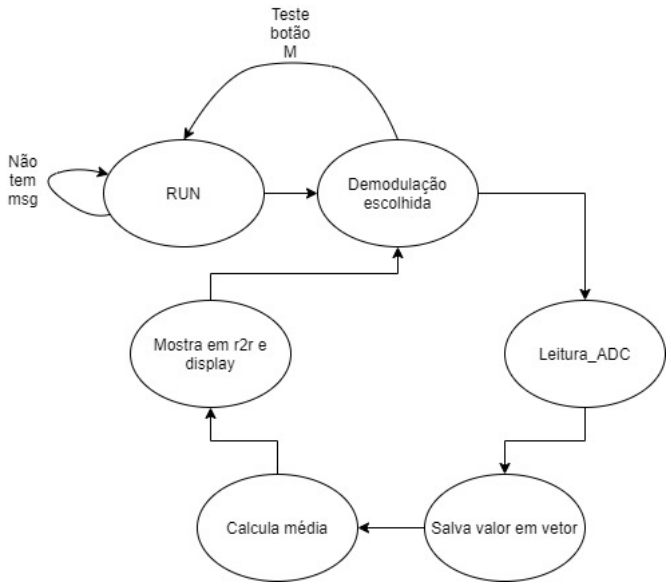
Fonte: Autoria própria

Na Figura 14 ocorre uma demodulação em ASK, verificou-se que o resultado apresenta um pequeno atraso, mas que não interfere na análise final, pois trata-se de um sinal digital, dessa forma, é fácil de distinguir o que refere -se à 1 ou ao 0. No gráfico *(a)*, tem-se o sinal digital, que deseja-se transmitir e em *(b)*, *(c)* e *(d)*, é o sinal sendo modulado, retificado e demodulado, respectivamente.

4.4.4 Algoritmo de demodulação

Na Figura 15 pode-se verificar como a demodulação é feita e como ela se comporta em relação a máquina de estado. Quando dentro de determinado estado de demodulação, então uma leitura é realizada, com esse dado é calculado o valor atual da média móvel, que posteriormente é exibido no r2r e no display no caso AM e apenas no display para casos de ASK.

Figura 15 – Fluxograma implementado em código para demodulação



Fonte: Autoria própria

Abaixo na Figura 16 vemos como o código foi implementado para demodulação AM. Após a leitura de tensão, devido ao offset, não podemos ter tensões negativas. Com isso, devemos identificar sim se o valor é inferior a 512 (representação do zero) ou superior, para assim retificar o valor. Em seguida se testa se já foram amostradas as 16 amostras para alocar os valores em todos os endereços do vetor. Caso tenha passado, o vetor será deslocado pra esquerda.

Em seguida todos os valores do vetor são somados e a média é calculada, posteriormente mostradas no display e R2R.

Figura 16 – Código implementado para demodulação

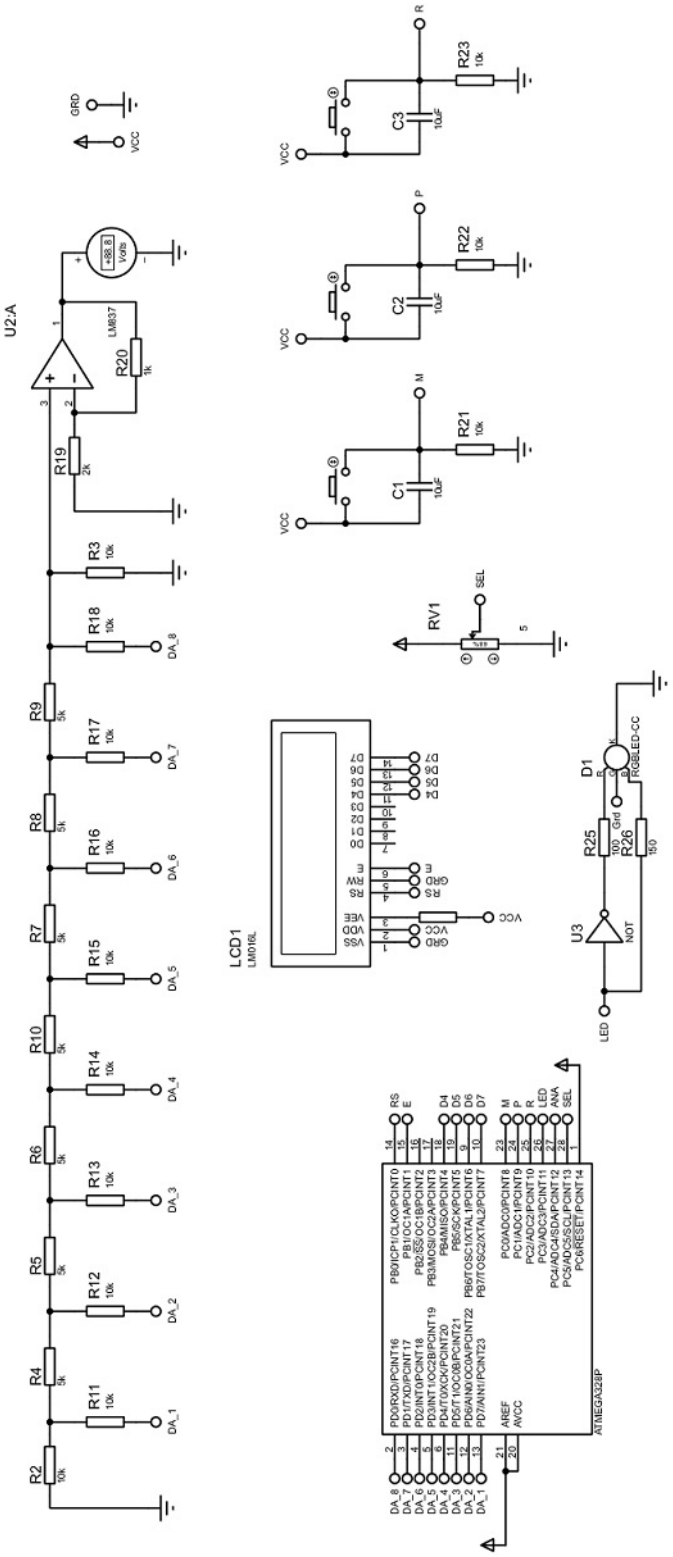
```
void demodAM()
{
    unsigned int temporario[16]={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, i=0, soma, demod_atual, decimal_atual;
    int j,k;
    while (BOTAO==FALSE) {
              soma=0;//reseta soma do valor acumulado
              leitura_adc();//Leitura do valor a ser amostrado
              if(resultado_adc<512)resultado_adc=1024-resultado_adc;//Testa se a tensão é negativa
              if(i<16){//Primeiras 16 amostras
                       temporario[i]=resultado_adc;
                       i++;
               }
               else{//Demais amostras
                       for(k=0;k<15;k++){
                            temporario[k]=temporario[k+1];
                       }
                       temporario[16]=resultado_adc;
               }
               for(j=0; i<16; i++){
                       soma=soma+temporario[j];//Soma das 16 últimas amostras
               }
               demod_atual = soma/16;//Calculo da média
               convert_r2r(demod_atual);//Função para converter amostragem nas entradas do R2R
               decimal_atual = binDec(demod_atual); //Conversão demod_atual em decimal
               show_demod(demod_atual); //Função para mostrar valor em LCD
               teste_botao();//Função para testar se botão M não foi pressionado
    }
}
```

Fonte: Autoria própria

5 Esquemático

Os componetes físicos necessários para as simulações foram, o AVR Atmega328P, Display LCD, circito do conversor DA, botões, Potenciômetro e do LED. A figura 17, apresenta como esses componetes conectam-se.

Figura 17 – Sistema físico do projeto



Fonte: Autoria própria

6 Resultados

Ao final de cada implementação de código, suas subrotinas foram testadas e com exceção das demodulações todas apresentaram êxito.

A configuração de periféricos não apresentaram grandes problemas em suas implementações. A problemática se apresentou no filtro.

Acreditamos que o problema está na conversão da mensagem analógica e em sua amostragem, que não apresenta uma resposta satisfatória de medição da tensão da entrada medida. Além disso, a quantidade de operações realizadas entre os períodos de amostragem aumenta o problema de amostragem do sistema, por contar com conjuntos de operações que exigem muito do ATMega. Sendo assim, a máquina não esta conseguindo amostrar devidamente o formato de sinal modulado. Ademais, o filtro utilizado ainda apresentou problemas na implementação de sua lógica aritmética, exigindo mais do que o esperado do AVR, devido a má implementação da formulação matemática do filtro escolhido.

7 Conclusão

Na semana anterior a esta implementação nos foi apresentado o desafio de projetar um rádio definido por software capaz de demodular mensagens enviadas e moduladas por quatro técnicas, AM, ASK, FM e FSK. Porém foram apresentadas dificuldades conceituais e o projeto foi reduzido para apenas AM e ASK.

A mudança para linguagem C troxe facilidades para implementações de lógica, visto que os próprios compiladores gerenciam a alocação de dados nos registradores, facilidades de operações aritméticas e manipulação de dados e instruções.

Ao longo da semana os conceitos de AM e ASK foram bem fixados entre os componentes, nos primeiros dias conseguimos implementar a lógica em matlab e colocar em prática algumas possibilidades de técnicas de filtragem, o que auxiliou muito no conhecimento teórico.

Devido a um mal gerenciamento de tempo, dedicamos muito tempo para a implementação mal sucedida em C da modulação AM, quando poderíamos ter tentado implementar o ASK, que acreditamos que seja mais simples. A mesma foi testada em matlab, mas não foi convertida para o C.

Referências

Borges de Lima, C; Villaça, M. V. M. AVR e arduino técnicas de projeto. Florianópolis, Brasil, Ed dos autores. 2012

FILTROS NA ANÁLISE CINEMÁTICA. Disponível em: <http://www.cpaqv.org/mtpmh/filtros_cinematica.pdf>. Acesso em 15 de Novembro de 2020

Modulação AM - técnico. Disponível em: <https://wiki.sj.ifsc.edu.br/index.php/Modula%C3%A7%C3%A3o_AM_-_t4%$C3\%A9cnico$ >. Acesso em 14 de Novembro de 2020

Interfacing LCD16x2 with AVR ATmega16/ATmega32 in 4-bit mode . Disponível em: <https://www.electronicwings.com/avr-atmega/interfacing-lcd-16x2-in-4-bit-mode-with-atmega-16-32->. Acesso em 13 de Novembro de 2020