DANIEL MADEIRA BUENO

Manual de implementação e simulação de projetos utilizando *Arduino UNO* e *Proteus/ISIS*

Acionamento de *SERVO MOTORES* e utilização de *DISPLAYS LCD* com o controlador *PCD8544*

Vitória – ES
NOV/2015

LISTA DE FIGURAS

SUMÁRIO

INTRODUÇÃO

O Proteus é um software criado pela empresa Labcenter Eletronics, este proporciona ao usuário a possibilidade de montar circuitos eletrônicos, realizar simulações e ainda criar placas de circuito impresso. Um dos componentes do Proteus é o ambiente de desenvolvimento ISIS que tem como função principal auxiliar o desenvolvedor de sistemas embarcados na verificação da parte eletrônica referente aos circuitos dos mesmos, além de possibilitar a realização de simulações (como dito anteriormente) antes de elaborar o hardware de um determinado projeto, podendo proporcionar portanto uma redução no tempo de desenvolvimento e até mesmo do custo envolvido.

POR QUE UTILIZAR O *PROTEUS/ISIS*?

Além dos motivos citados anteriormente, para muitas pessoas não é possível desenvolver todas as ideias que surgem na cabeça por vários motivos, um deles pode ser a falta de acesso aos materiais e equipamentos para o criação de protótipos e realização de testes. Por outro lado, existem ferramentas criadas para justamente suprir esta carência, entre elas está o *Proteus/ISIS*, este conta com diversas ferramentas que estão presentes em laboratórios de eletrônica como osciloscópio, gerador de sinais, voltímetro, amperímetro, além de uma infinidade de componentes como resistores, capacitores, transistores, motores, microcontroladores e outros (muitos outros).

QUAL O OBJETIVO DESTE DOCUMENTO?

Este documento visa ao proporcionar o aprendizado básico para a manipulação deste software bem como uma maneira bastante didática e simples de como realizar pequenos projetos envolvendo a utilização do *Arduino UNO* para aqueles que desejam aprender um pouco sobre o assunto e no momento não possuem conhecimento algum sobre o mesmo. Obviamente existem limitações no que diz respeito à implementação de projetos utilizando o *Proteus/ISIS* (no decorrer dos projetos apresentados serão discutidas algumas limitações caso estas sejam importantes no momento), no entanto, ainda é possível realizar bastante coisa.

O QUE ESTE DOCUMENTO ABRANGE?

Num primeiro momento apresenta-se um pouco da interface do *Proteus/ISIS* e como fazer para selecionar componentes, instalar bibliotecas de componentes, definir propriedades de componentes entre outros.

Posteriormente será mostrado como fazer para importar o código desenvolvido para dentro do *Proteus/ISIS* e simula-lo em um *Arduino UNO* virtual.

Serão desenvolvidos também vários projetos básicos organizados para que exista uma certa progressão no conhecimento adquirido para criação dos mesmos, facilitando portanto, o aprendizado de maneira bastante simples. Neste documento encontram-se projetos envolvendo *displays LCD* que utilizam o PCD8544 como controlador e servo motores.

SOBRE O AUTOR

Meu nome é Daniel Madeira Bueno, sou engenheiro eletricista graduado pela Universidade Federal do Espírito Santo. Gosto bastante de questões que envolvem sistemas embarcados porém não sou profissional no assunto, portanto organizei este conteúdo da forma que geralmente eu consigo aprender melhor e resolvi compartilhar. Se possível peço que caso existam sugestões, pontos para corrigir ou qualquer outra coisa, entre em contato comigo, ficarei extremamente agradecido em poder aprimorar meu conhecimento e ajudar os outros da mesma forma.

Meu e-mail é daniel_m_bueno@hotmail.com

ACIONAMENTO DE SERVO MOTORES E UTILIZAÇÃO DE DISPLAYS LCD COM O CONTROLADOR PCD8544

Neste parte serão apresentados alguns projetos básicos utilizando *displays LCD* funcionando através do controlador PCD8544, proporcionando a possibilidade de aprendizado sobre um outro *display* além do descrito no material anterior. Além disso serão exploradas algumas situações onde pode-se aprender um pouco sobre o acionamento de servo motores. Este material é uma sequência de dois outros publicados anteriormente relacionados ao uso de *leds* e *displays LCD* com o controlador *HD44780*, portanto é aconselhável utilizar os três materiais juntos para consolidar de maneira mais eficiente o conhecimento adquirido.

1 PRIMEIROS PASSOS COM O DISPLAY NOKIA 5110

1.1 Objetivo

Este item será o ponto de partida para os estudos propostos na terceira parte deste grande material sobre simulação do *Arduino UNO* no *Proteus/ISIS.* Primeiramente é realizada uma primeira apresentação sobre o *display NOKIA 5110* (manipulado através do controlador *PCD8544*) juntamente com os procedimentos preliminares para a utilização do mesmo bem como um exemplo introdutório mostrando como fazer para escrever algo no *display.*

1.2 Hardware utilizado na simulação

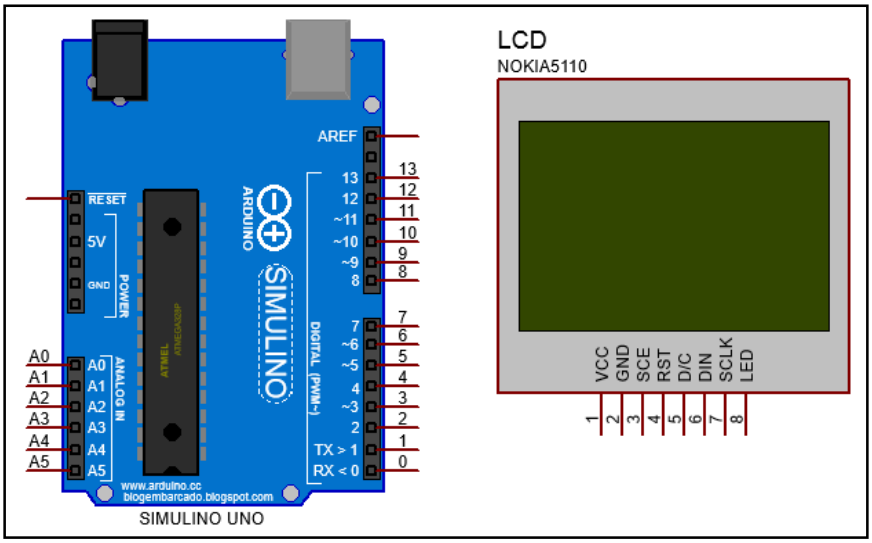


Figura 1 - Hardware utilizado no item 1

Assim como foi realizado no material anterior referente à simulação do *Arduino UNO* no *Proteus/ISIS*, neste momento também serão desenvolvidas aplicações utilizando displays LCD, no entanto, o alvo desta parte será o *Nokia 5110* manipulado através do controlador *PCD8544.* O componente em questão é um display monocromático com resolução de 84x48 pixels e uma tensão de operação 3.3V ou 5V dependendo do modelo físico (as simulações serão feitas alimentando o *display* com 5V, em virtude de não ser necessária a utilização de meios para reduzir esta tensão).

1.3 Diagrama esquemático das ligações

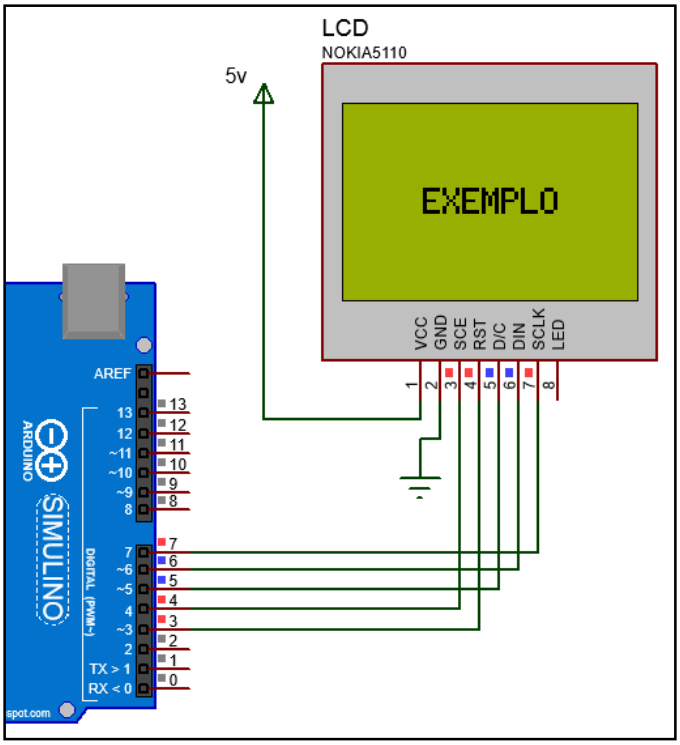


Figura 2 – Diagrama esquemático utilizado no item 1

1.4 Desenvolvimento do código

Atualmente existem algumas bibliotecas que podem servir para manipular o *display Nokia 5110*, no entanto, serão utilizadas três bibliotecas em conjunto para o controle do *display*. Estas são:

- SPI.h
- Adafruit_PCD8544.h
- Adafruit_GFX.h

A primeira entre as três bibliotecas citadas, já se encontra previamente "instalada", porém para a utilização das outras duas deve-se realizar alguns procedimentos antes de simplesmente declara-las no código. Primeiramente é necessário baixar ambas (disponibilizadas pela Adafruit) nos seguintes links:

- https://github.com/adafruit/Adafruit-PCD8544-Nokia-5110-LCD-library
- https://github.com/adafruit/Adafruit-GFX-Library

Após adquirir as duas bibliotecas, basta descompactar o conteúdo dentro da pasta *Libraries* (localizada onde está instalada a *IDE* do *Arduino)* e alterar os nomes das pastas para *Adafruit_PCD8544* e *Adafruit_GFX* respectivamente.

No decorrer deste tópico serão explicados todos os passos utilizados para o desenvolvimento do projeto, isto é, o código será comentado de modo que todo conhecimento aplicado seja facilmente compreendido. As trechos demarcados em vermelho contém elementos que não foram citados anteriormente ou são partes de bastante relevância para o funcionamento do programa. O código inteiro está disponibilizado no apêndice A, localizado no final deste documento.

Primeiramente deve-se incluir as bibliotecas necessárias para utilizar o display através do comando *#include* seguido do nome das bibliotecas (pode ser representada de duas maneiras: <biblioteca.h>, "biblioteca.h"). Estas são compostas por uma série de códigos que visam auxiliar, tornando mais simples a interação do *Arduino UNO* com o controlador do *display* utilizado (PCD8544). A biblioteca SPI.h é responsável pela utilização do protocolo SPI para a manipulação dos dados enquanto as bibliotecas *Adafruit_PCD8544.h* e *Adafruit_GFX.h* proporcionam a facilidade no acionamento do *display* e uma variedade de funções relacionadas à produção elementos gráficos respectivamente.

```
#include <SPI.h>
#include <Adafruit_PCD8544.h>
#include <Adafruit_GFX.h>
```

O Segundo passo que deve ser realizado antes da manipulação do *display* é a criação de um objeto do tipo Adafruit_PCD8544 cujo nome é ***lcd***. Observe que são passados alguns valores como parâmetros da função, estes são utilizados para relacionar os pinos do *Arduino UNO* com os pinos do controlador. Existem várias maneiras de criar o objeto do tipo Adafruit_PCD8544 (no que diz respeito à quantidade de parâmetros a serem utilizados), neste caso foi utilizada uma declaração com cinco parâmetros e estes podem ser identificados como:

- ***Adafruit_PCD8544 (SCLK, DIN, DC, CS, RST)***

```
Adafruit_PCD8544 lcd = Adafruit_PCD8544(7, 6, 5, 4, 3);
```

Em seguida, deve-se realizar apenas mais um procedimento para então começar a manipular o *display*. Dentro da função ***setup()*** foi utilizada a função ***begin()*** responsável por iniciar a interface de comunicação entre o *Arduino UNO* e o *display*. Neste momento torna-se possível a utilização do *display* no desenvolvimento do projeto.

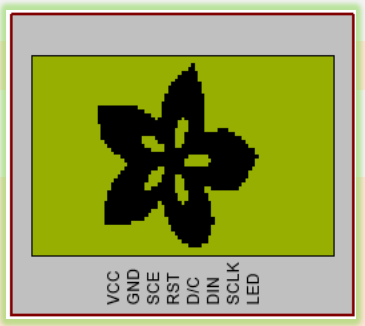
Assim que a função ***begin()*** é utilizada, gera-se uma animação com o logotipo da *Adafruit*. Obviamente existem meios de remover esta animação, no entanto, esta será preservada em forma de agradecimento e reconhecimento do trabalho feito com intenção de facilitar nosso desenvolvimento de projetos, sendo assim, esta é mantida durante dois segundos até que é utilizada a função ***clearDisplay(),*** responsável por apagar o *display*.

Posteriormente são utilizadas algumas funções para configurar o que virá a ser escrito em seguida. A função ***setTextSize()*** é responsável pelo tamanho da fonte (o argumento desta função pode ser 1,2 ou 3, de modo que quanto maior, maior será a fonte), já a função ***setTextColor()*** determina a cor da fonte (o argumento pode ser BLACK ou WHITE, determinando portanto se a mesma é preta ou branca). Por último, a função ***setCursor()*** posiciona o cursor no pixel escolhido (desta vez o primeiro argumento está ligado à coordenada do eixo x e o segundo diz respeito à coordenada do eixo y).

Para imprimir alguma informação no *display* basta utilizar a função ***print()*** e passar a mensagem desejada como argumento da mesma (entre aspas). Um ponto muito importante deve ser destacado: Sempre que forem utilizadas funções pertinentes ao funcionamento do *display* deve-se preceder as mesmas do objeto (do tipo Adafruit_PCD8544) criado, por exemplo ***lcd.print("EXEMPLO")*.**

O ponto mais importante deste exemplo está na função *display()*, pois esta é responsável pela execução de todos as funções anteriores, ou seja, todas as alterações que são feitas no *display*, só acontecem após a utilização da função *display()*.

```
void setup() {

  lcd.begin();
  delay(2000);

  lcd.clearDisplay();

  lcd.setTextSize(1);
  lcd.setTextColor(BLACK);
```



```
    lcd.setCursor(22,20);
    lcd.print("EXEMPLO");

    lcd.display()
}
```

Neste item não houve a necessidade de utilizar a função ***loop().***

```
void loop() {

}
```

2 ACIONAMENTO BÁSICO DE UM SERVO MOTOR

2.1 Objetivo

Os objetivos deste item são basicamente proporcionar o conhecimento necessário para a realização do acionamento básico de um servo motor, além de desenvolver um código que permita informe de maneira bastante simplista o ângulo quem que este irá parar através do *display* alvo deste material.

2.2 Hardware utilizado na simulação

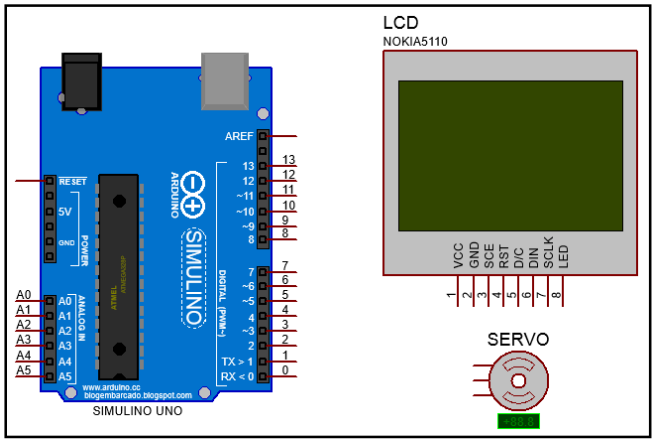


Figura 3 - Hardware utilizado no item 2

2.3 Diagrama esquemático das ligações

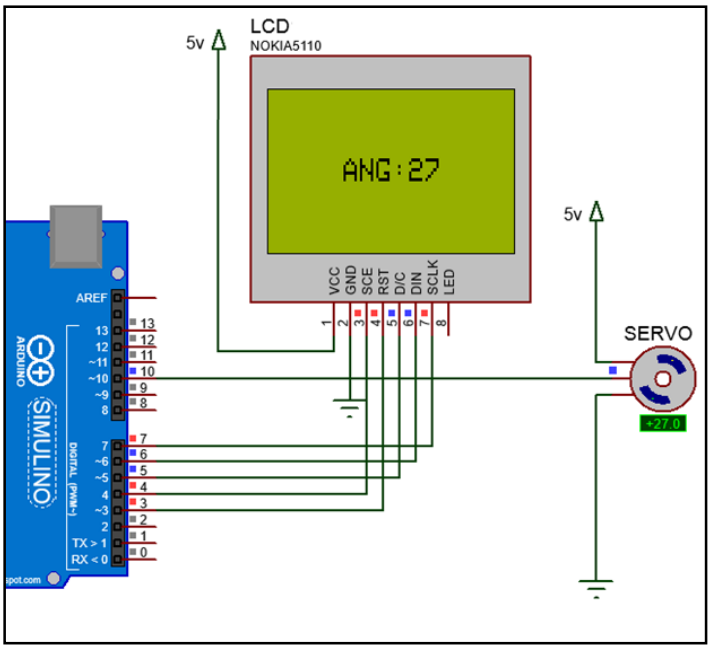
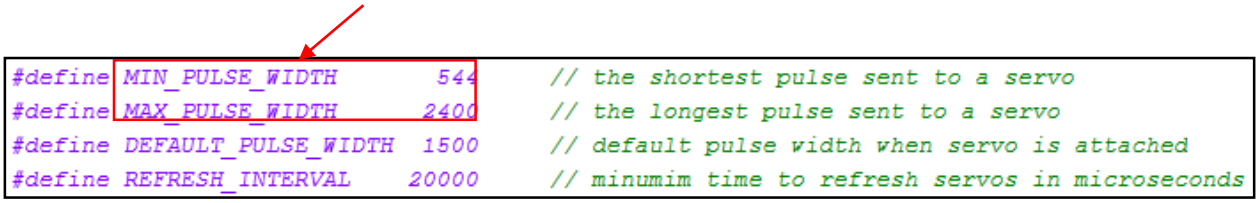


Figura 4 - Diagrama esquemático utilizado no item 2

2.4 Desenvolvimento do código

Antes de começarmos no estudo da manipulação dos servo motores, deve-se realizar alguns procedimentos para que a simulação seja feita de maneira correta. Repare que a biblioteca ***servo.h*** (biblioteca que será utilizada neste exemplo) foi elaborada para trabalhar com servo motores que possuem valores de alguns parâmetros previamente determinados, de modo que é necessário que estes parâmetros estejam de acordo com os parâmetros do motor no Proteus/ISIS. Deve-se proceder da seguinte maneira para realizar as modificações necessárias.



```
#define MIN_PULSE_WIDTH       544     // the shortest pulse sent to a servo
#define MAX_PULSE_WIDTH      2400     // the longest pulse sent to a servo
#define DEFAULT_PULSE_WIDTH  1500     // default pulse width when servo is attached
#define REFRESH_INTERVAL    20000     // minumim time to refresh servos in microseconds
```

Figura 5 – Trecho da biblioteca servo.h

Para realizar as modificações necessárias basta entrar na janela de preferências do servo e alterar os seguintes itens (os tempos estão em microssegundos):

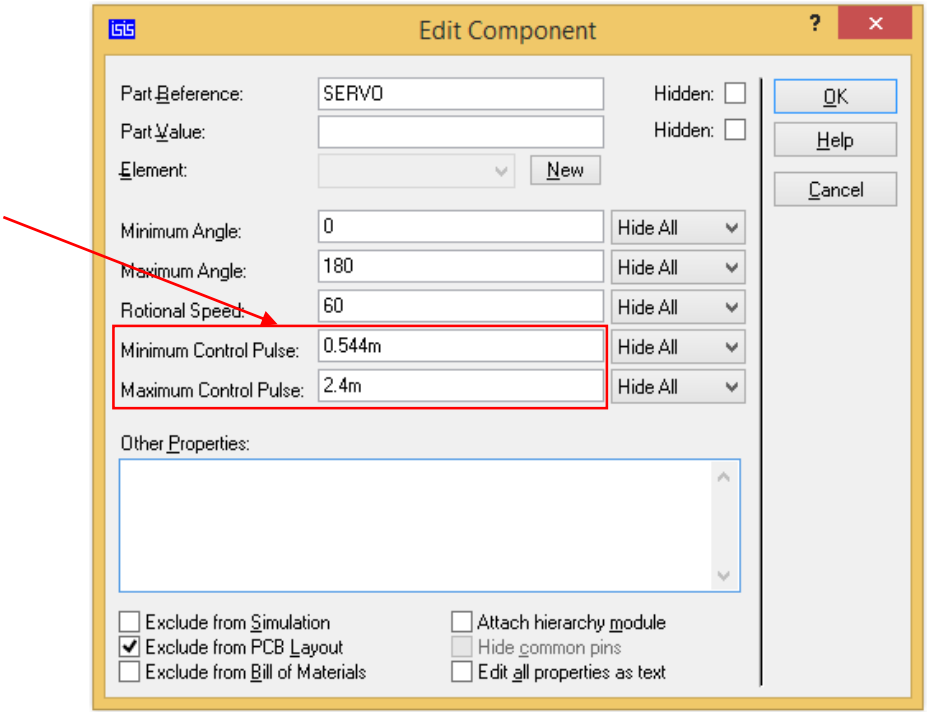


Figura 6 – Propriedades do servo motor

No decorrer deste tópico serão explicados todos os passos utilizados para o desenvolvimento do projeto, isto é, o código será comentado de modo que todo conhecimento aplicado seja facilmente compreendido. As trechos demarcados em vermelho contém elementos que não foram citados anteriormente ou são partes de bastante relevância para o funcionamento do programa. O código inteiro está disponibilizado no apêndice A, localizado no final deste documento.

Primeiramente deve-se incluir as bibliotecas necessárias para utilizar o *display* e o servo motor, pois estas são compostas por uma série de códigos que visam auxiliar, tornando mais simples a interação do *Arduino UNO* com o controlador do *display* utilizado (PCD8544) e com o servo motor. A biblioteca ***SPI.h*** é responsável pela utilização do protocolo SPI para a manipulação dos dados, as bibliotecas ***Adafruit_PCD8544.h*** e ***Adafruit_GFX.h*** proporcionam a facilidade no acionamento do *display* e uma variedade de funções relacionadas à produção elementos gráficos respectivamente enquanto a biblioteca ***servo.h*** é necessária para o acionamento do servo motor.

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>
#include <Servo.h>
```

O Segundo passo que deve ser realizado antes da manipulação dos elementos existentes no sistema é a criação de um objeto do tipo Adafruit_PCD8544 (como explicado no item anterior) cujo nome é ***lcd*** e a criação de um objeto do tipo Servo identificado por ***servo***, utilizado para a manipulação do motor.

```
Adafruit_PCD8544 lcd = Adafruit_PCD8544(7, 6, 5, 4, 3);
Servo servo;
```

Posteriormente percebe-se que existe uma variável chamada ***angulo***, responsável por conter o valor desejado da rotação do servo motor.

```
  int angulo= 27;
```

Além de utilizar a função a função ***begin()*** dentro da função ***setup()*** deve-se realizar outro procedimento antes da manipulação dos elementos, este consiste em utilizar a função ***attach()*** para que o *Arduino UNO* saiba em qual pino o servo motor está conectado. Desta maneira torna-se possível a utilização do *display* e do servo motor no desenvolvimento do projeto.

Em seguida são utilizadas algumas funções relativas à escrita no *display*. O interessante neste momento é que o cursor é posicionado através da função ***setCursor()*** e escreve-se no display "ANG:" utilizando a função ***print()***, note que caso algo seja escrito sem que o cursor seja novamente posicionado em algum ponto, este partirá seguidamente ao que foi escrito anteriormente, logo, utilizando mais uma vez a função ***print()*** porém tendo a variável ***angulo*** como parâmetro, torna-se possível representar o valor da mesma no *display*

Por fim o servo deve realizar uma rotação de 27 graus, esta ação é determinada pela função ***write()***.

```
void setup() {

    servo.attach(10);

    lcd.begin();
    delay(2000);
    lcd.clearDisplay();

    lcd.setTextSize(1);
    lcd.setTextColor(BLACK);
    lcd.setCursor(25,20);
    lcd.print("ANG:");
    lcd.print(angulo);

    lcd.display();

    servo.write(angulo);

}
```

Neste item não houve a necessidade de utilizar a função ***loop().***

```
void loop() {

}
```

3 ACIONAMENTO DE SERVOS UTILIZANDO POTENCIÔMETROS

3.1 Objetivo

Neste item será realizado o acionamento de quatro servo motores por meio de potenciômetros, também será desenvolvida nesta seção uma tela informativa que será apresentada no *display* utilizado.

3.2 Hardware utilizado na simulação

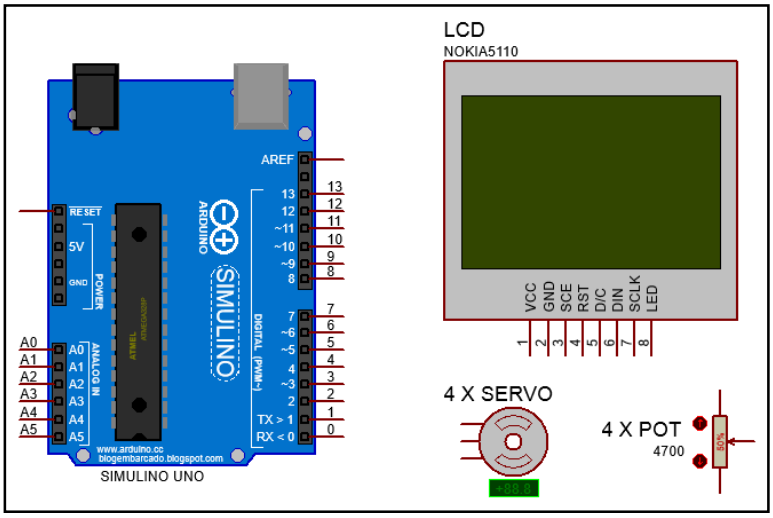


Figura 7 - Hardware utilizado no item 3

3.3 Diagrama esquemático das ligações

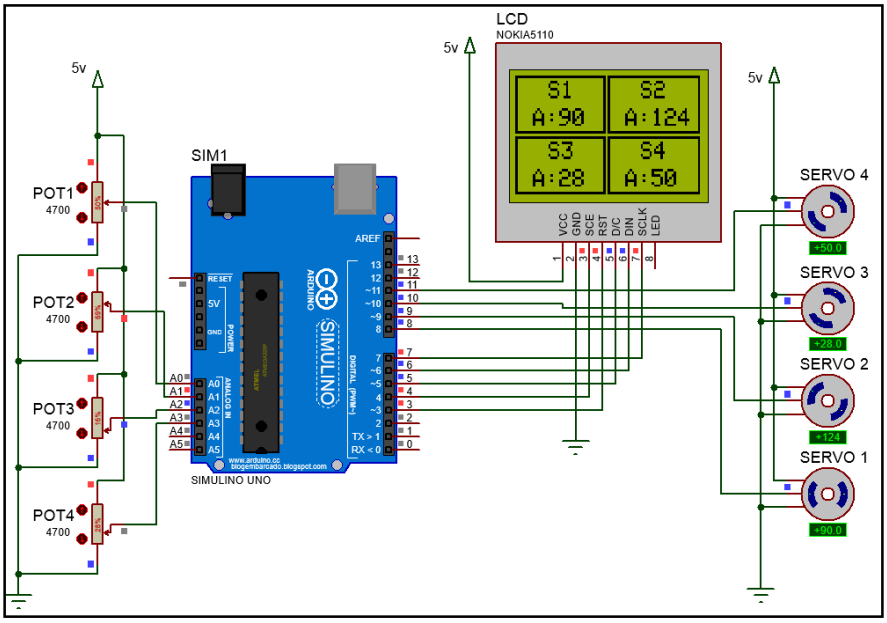


Figura 8 - Diagrama esquemático utilizado no item 3

3.4 Desenvolvimento do código

No decorrer deste tópico serão explicados todos os passos utilizados para o desenvolvimento do projeto, isto é, o código será comentado de modo que todo conhecimento aplicado seja facilmente compreendido. As trechos demarcados em vermelho contém elementos que não foram citados anteriormente ou são partes de bastante relevância para o funcionamento do programa. O código inteiro está disponibilizado no apêndice A, localizado no final deste documento.

Primeiramente deve-se incluir as bibliotecas necessárias para utilizar o *display* e o servo motor, pois estas são compostas por uma série de códigos que visam auxiliar, tornando mais simples a interação do *Arduino UNO* com o controlador do *display* utilizado (PCD8544) e com o servo motor. A biblioteca ***SPI.h*** é responsável pela utilização do protocolo SPI para a manipulação dos dados, as bibliotecas ***Adafruit_PCD8544.h*** e ***Adafruit_GFX.h*** proporcionam a facilidade no acionamento do *display* e uma variedade de funções relacionadas à produção elementos gráficos respectivamente enquanto a biblioteca ***servo.h*** é necessária para o acionamento do servo motor.

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>
#include <Servo.h>
```

Em seguida utiliza-se a diretiva *#define* para associarmos os elementos aos seus respectivos pinos. Neste momento os potenciômetros 1,2,3 e 4 estão relacionados com os pinos de entrada analógica A0,A1,A2 e A3 respectivamente.

```
#define POT1 0
#define POT2 1
#define POT3 2
#define POT4 3
```

Posteriormente são criados os objetos do tipo Adafruit_PCD8544 cujo nome é ***lcd*** e quatro objetos do tipo Servo identificados por ***servo1/2/3/4***, utilizados para a manipulação dos motores. Além disso, são declaradas as variáveis ***medidaPOT1/POT2/POT3/POT4*** responsáveis por armazenar os valores atuais

obtidos pelos potenciômetros através da função ***analogRead()***, já as variáveis ***medidaPOT1ant/POT2ant/ POT3ant/POT4ant*** guardam os valores anteriores obtidos pelos potenciômetros.

```
Adafruit_PCD8544 lcd = Adafruit_PCD8544(7, 6, 5, 4, 3);

Servo servo1;
Servo servo2;
Servo servo3;
Servo servo4;

int medidaPOT1;
int medidaPOT1ant;

int medidaPOT2;
int medidaPOT2ant;

int medidaPOT3;
int medidaPOT3ant;
int medidaPOT4;
int medidaPOT4ant;
```

Dentro da função ***setup()*** existem apenas duas funções. A função ***iniciarSERVOS()*** é responsável por atribuir os servos aos respectivos pinos através da função ***attach()*** além de mapear os potenciômetros e posicionar os motores no ângulo inicial de acordo com os mesmos e atualizar o valor das variáveis que registram os valores anteriores já citadas.

A segunda função presente na função ***setup()*** é a ***iniciarLCD()***, em primeiro lugar, é inicializada a interface entre o *display* e o *Arduino UNO* através da função ***begin()***, em seguida são desenhados na tela quadro retângulos por meio da função ***drawRect()*** para que as informações de cada servo sejam disponibilizadas de forma visualmente organizada (observe que esta função possui cinco parâmetros, onde os dois primeiros são referentes ao ponto do canto superior esquerdo do retângulo, o terceiro e o quarto estão relacionados com o comprimento e altura respectivamente, enquanto o último parâmetro define a cor da linha utilizada na criação dos mesmos.

Posteriormente, os quatro campos criados são identificados com as siglas S1/S2/S3 e S4 e também são escritos os valores correspondentes dos ângulos iniciais de cada servo.

```
void setup() {

  iniciarSERVOS();
  iniciarLCD();

}
```

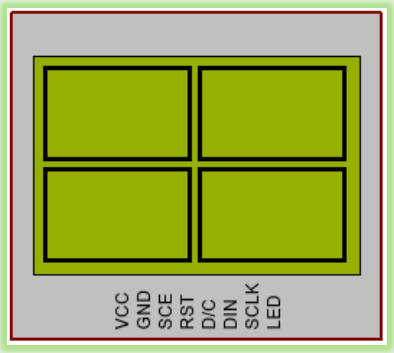
Dentro da função ***loop()*** está presente a função ***atualizarSERVOS()***, esta é responsável por realizar a leitura dos potenciômetros, converter para um intervalo entre 0 e 180 graus através da função ***map()*** e tomar as providencias necessárias caso a leitura atual seja diferente da anterior (estas providencias são: a atualização do display com o valor atual e a movimentação de um determinado motor para o valor desejado).

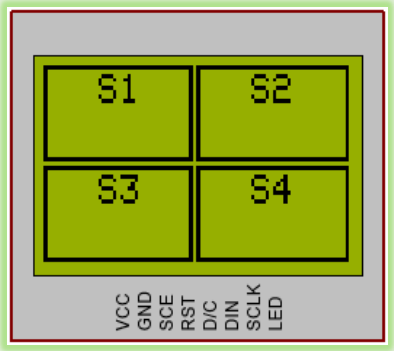
Deve-se ressaltar que este *display* funciona de maneira diferente do apresentado no material anterior, pois não é possível simplesmente escrever um caractere sobre outro de maneira em que o primeiro desapareça, então neste caso, as partes relativas aos valores dos ângulos (que são as que mudam de acordo com o funcionamento do programa) são reescritas na cor "branca" (para "apagar") e só depois deste passo são escritos os valores atuais na cor "preta".

```
void loop() {

  atualizarSERVOS();

}
```

```
void iniciarLCD(){

    lcd.begin();
    delay(2000);
    lcd.clearDisplay();
```

```
lcd.drawRect(0,0,41, 23, BLACK);
lcd.drawRect(42,0,42, 23, BLACK);
lcd.drawRect(0,24,41, 23, BLACK);
lcd.drawRect(42,24,42, 23, BLACK);

lcd.setTextSize(1);
lcd.setTextColor(BLACK);
lcd.setCursor(15,2);
lcd.print("S1");

lcd.setCursor(57,2);
lcd.print("S2");

lcd.setCursor(15,26);
lcd.print("S3");

lcd.setCursor(57,26);
lcd.print("S4");
lcd.setCursor(2,13);
lcd.print(" A:");
lcd.print(medidaPOT1);

lcd.setCursor(44,13);
lcd.print(" A:");
lcd.print(medidaPOT2);

lcd.setCursor(2,37);
lcd.print(" A:");
lcd.print(medidaPOT3);

lcd.setCursor(44,37);
lcd.print(" A:");
lcd.print(medidaPOT4);
```

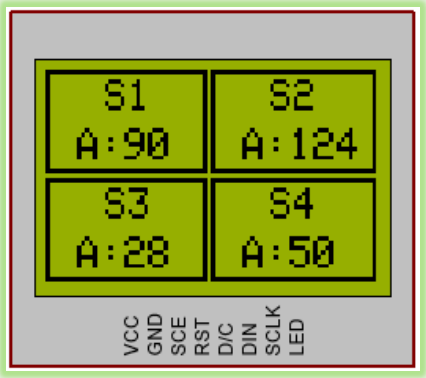




```
    lcd.display();

}
```

```
void iniciarSERVOS(){

    servo1.attach(8);
    servo2.attach(9);
    servo3.attach(10);
    servo4.attach(11);



    medidaPOT1 = analogRead(POT1);
    medidaPOT1 = map(medidaPOT1,0,1023,0,180);

    medidaPOT2 = analogRead(POT2);
    medidaPOT2 = map(medidaPOT2,0,1023,0,180);

    medidaPOT3 = analogRead(POT3);
    medidaPOT3 = map(medidaPOT3,0,1023,0,180);

    medidaPOT4 = analogRead(POT4);
    medidaPOT4 = map(medidaPOT4,0,1023,0,180);

    servo1.write(medidaPOT1);
    servo2.write(medidaPOT2);
    servo3.write(medidaPOT3);
    servo4.write(medidaPOT4);

    medidaPOT1ant = medidaPOT1;
    medidaPOT2ant = medidaPOT2;
    medidaPOT3ant = medidaPOT3;
    medidaPOT4ant = medidaPOT4;
```

```
}

void atualizarSERVOS(){

    medidaPOT1 = analogRead(POT1);
    medidaPOT1 = map(medidaPOT1,0,1023,0,180);

     if (medidaPOT1 != medidaPOT1ant){

          lcd.setTextSize(1);
          lcd.setCursor(2,13);
          lcd.print(" A:");
          lcd.setTextColor(WHITE);
          lcd.print(medidaPOT1ant);
          lcd.setTextColor(BLACK);
          lcd.setCursor(2,13);
          lcd.print(" A:");
          lcd.print(medidaPOT1);
          lcd.display();

          medidaPOT1ant = medidaPOT1;
          servo1.write(medidaPOT1);
    }

    medidaPOT2 = analogRead(POT2);
    medidaPOT2 = map(medidaPOT2,0,1023,0,180);

    if (medidaPOT2 != medidaPOT2ant){

          lcd.setTextSize(1);
          lcd.setCursor(44,13);
          lcd.print(" A:");
          lcd.setTextColor(WHITE);
          lcd.print(medidaPOT2ant);
```



```
        lcd.setTextColor(BLACK);
        lcd.setCursor(44,13);
        lcd.print(" A:");
        lcd.print(medidaPOT2);
        lcd.display();

        medidaPOT2ant = medidaPOT2;
        servo2.write(medidaPOT2);
    }

    medidaPOT3 = analogRead(POT3);
    medidaPOT3 = map(medidaPOT3,0,1023,0,180);

    if (medidaPOT3 != medidaPOT3ant){

        lcd.setTextSize(1);
        lcd.setCursor(2,37);
        lcd.print(" A:");
        lcd.setTextColor(WHITE);
        lcd.print(medidaPOT3ant);
        lcd.setTextColor(BLACK);
        lcd.setCursor(2,37);
        lcd.print(" A:");
        lcd.print(medidaPOT3);
        lcd.display();

        medidaPOT3ant = medidaPOT3;
        servo3.write(medidaPOT3);
    }

     medidaPOT4 = analogRead(POT4);
     medidaPOT4 = map(medidaPOT4,0,1023,0,180);

     if (medidaPOT4 != medidaPOT4ant){
```

```
        lcd.setTextSize(1);
        lcd.setCursor(44,37);
        lcd.print(" A:");
        lcd.setTextColor(WHITE);
        lcd.print(medidaPOT4ant);
        lcd.setTextColor(BLACK);
        lcd.setCursor(44,37);
        lcd.print(" A:");
        lcd.print(medidaPOT4);
        lcd.display();

        medidaPOT4ant = medidaPOT4;
        servo4.write(medidaPOT4);
    }
}
```

4 ACIONAMENTO DE UM SERVO MOTOR COM VELOCIDADE VARIÁVEL

4.1 Objetivo

O objetivo deste item consiste em propor um sistema composto de um menu básico feito no *display* possibilitando que através de botões o usuário possa definir o ângulo que deseja rotacionar o eixo do motor bem como a velocidade em que este processo será realizado, além disso estará disponível também a possibilidade de que o motor possa parar em qualquer posição e ao retomar seu funcionamento possa ir para o destino final anterior ou para um novo ângulo final.

4.2 Hardware utilizado na simulação

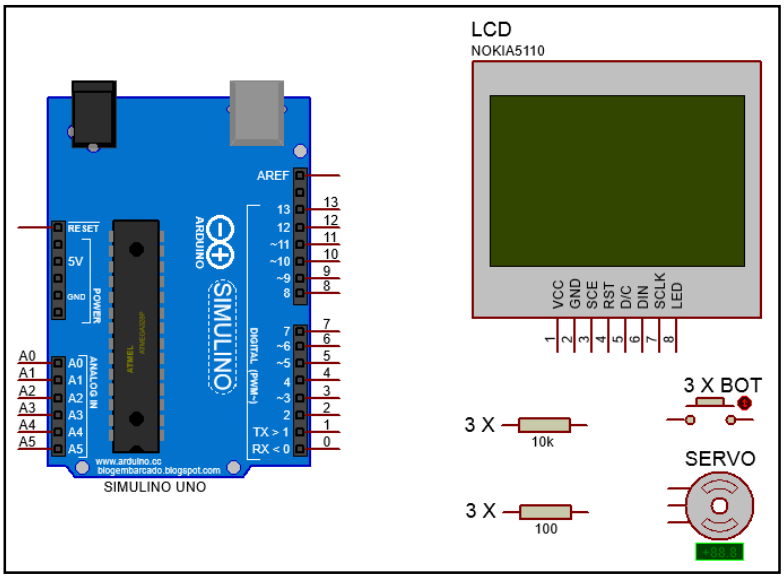


Figura 9 - Hardware utilizado no item 4

4.3 Diagrama esquemático das ligações

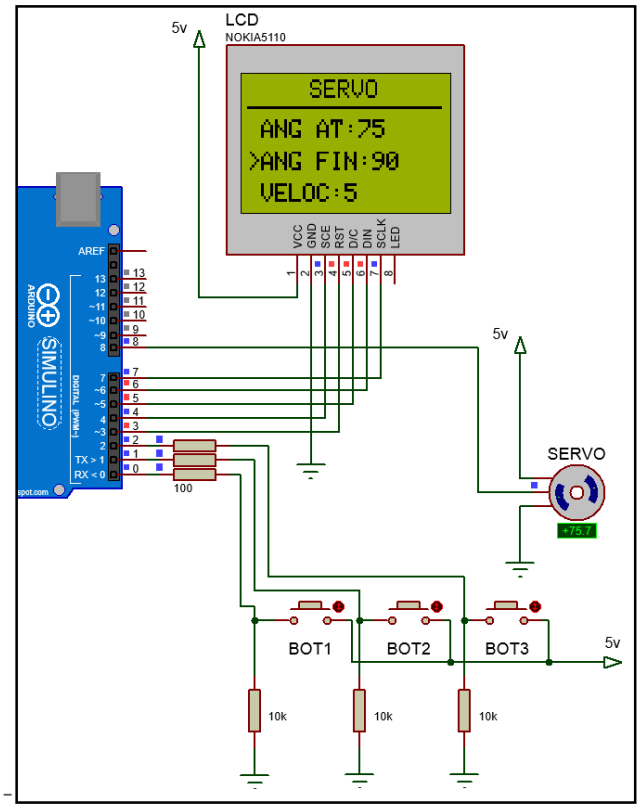


Figura 10 - Diagrama esquemático utilizado no item 4

4.3 Desenvolvimento do código

No decorrer deste tópico serão explicados todos os passos utilizados para o desenvolvimento do projeto, isto é, o código será comentado de modo que todo conhecimento aplicado seja facilmente compreendido. As trechos demarcados em vermelho contém elementos que não foram citados anteriormente ou são partes de bastante relevância para o funcionamento do programa. O código inteiro está disponibilizado no apêndice A, localizado no final deste documento.

Primeiramente deve-se incluir as bibliotecas necessárias para utilizar o *display* e o servo motor, pois estas são compostas por uma série de códigos que visam auxiliar, tornando mais simples a interação do *Arduino UNO* com o controlador do *display* utilizado (PCD8544) e com o servo motor. A biblioteca ***SPI.h*** é responsável pela utilização do protocolo SPI para a manipulação dos dados, as bibliotecas ***Adafruit_PCD8544.h*** e ***Adafruit_GFX.h*** proporcionam a facilidade no acionamento do *display* e uma

variedade de funções relacionadas à produção elementos gráficos respectivamente enquanto a biblioteca ***VarSpeedServo.h*** é necessária para o acionamento do servo motor.

A biblioteca ***VarSpeedServo.h*** pode ser baixada em: https://github.com/netlabtoolkit/VarSpeedServo. Com esta biblioteca é possível realizar uma infinidade de acionamentos, programar sequências e inclusive manter o programa rodando sem que o motor tenha acabado de realizar sua rotação

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>
#include <VarSpeedServo.h>
```

Em seguida utiliza-se a diretiva *#define* para associarmos os elementos aos seus respectivos pinos. Neste momento os botões 1(BOT1), 2(BOT2) e 3(BOT3) estão relacionados com os pinos 0,1 e 2 respectivamente.

```
#define BOT1 0
#define BOT2 1
#define BOT3 2
```

Posteriormente são criados os objetos do tipo Adafruit_PCD8544 cujo nome é ***lcd*** e um objeto do tipo VarSpeedServo identificado por ***servo***, utilizado para a manipulação do motor. Além disso num primeiro momento, são declaradas as variáveis que atuarão diretamente no acionamento do servo motor: a variável ***estadoSERVO*** é responsável pelo acionamento e parada do motor, a variável ***velocidade*** armazena a informação relativa a qual velocidade o motor deverá ter ao rotacionar, já as variáveis ***anguloANTERIOR/ATUAL/FIN*** estão diretamente relacionadas às posições do motor.

Em seguida está o bloco cujas variáveis são responsáveis pelo funcionamento dos três botões existentes no hardware da simulação. Observe que os três botões possuem tratamento em seu acionamento (Se o leitor tive alguma dúvida sobre a implementação do tratamento em botões, favor consultar o item 2.1 da primeira parte). Por fim temos as variáveis ***linhaATUAL*** e ***linhaANTERIOR***, utilizadas no funcionamento do menu apresentado pelo *display LCD*.

```
Adafruit_PCD8544 lcd = Adafruit_PCD8544(7, 6, 5, 4, 3);
VarSpeedServo servo;

bool estadoSERVO = 0;
int velocidade = 0;

int anguloANTERIOR=0;
int anguloATUAL=0;
int anguloFIN = 0;

bool estadoatualBOT1 = 0;
bool estadoantBOT1 = 0;
bool leituraBOT1 = 0;
long ultDebounceBOT1 = 0;


bool estadoatualBOT2 = 0;
bool estadoantBOT2 = 0;
bool leituraBOT2 = 0;
long ultDebounceBOT2 = 0;

bool estadoatualBOT3 = 0;
bool estadoantBOT3 = 0;
bool leituraBOT3 = 0;
long ultDebounceBOT3 = 0;

long tempoDeb = 50;

bool linhaATUAL;
bool linhaANTERIOR;
```

Dentro da função ***setup()*** serão definidos os modos de operação dos pinos utilizados, onde os pinos em que estão conectados os três botões são definidos como entradas. Neste momento também associa-se o servo ao pino 8 através da função ***attach()*** e este é colocado em uma posição de 0 graus utilizando a função ***write()***.

Em seguida, a interface entre o *display* e o *Arduino UNO* é inicializada com o uso da função ***begin()***. Logo após esta inicialização, é desenvolvida um menu, onde o usuário receberá a informação relativa ao ângulo atual do servo e ainda poderá optar por variar duas grandezas, o ângulo final que o motor deve atingir e a velocidade com que fará esta rotação.

```
void setup() {

    pinMode(BOT1,INPUT);
    pinMode(BOT2,INPUT);
    pinMode(BOT3,INPUT);

    servo.attach(8);
    servo.write(0);

    lcd.begin();
    delay(2000);
    lcd.clearDisplay();

    lcd.setTextSize(1);
    lcd.setTextColor(BLACK);
    lcd.setCursor(27,0);
    lcd.print("SERVO");
    lcd.drawLine(0, 10, 84, 10, BLACK);

    lcd.setCursor(0,15);
    lcd.print(" ANG AT:0");

    lcd.setCursor(0,27);
    lcd.print(">ANG FIN:0");

    lcd.setCursor(0,39);
    lcd.print(" VELOC:0");

    lcd.display();

}
```

Dentro da função ***loop()*** existem cinco funções responsáveis por promover a verificação de alguns elementos, primeiramente, a função ***verificarBOT1()*** analisa o estado do botão um para saber se este foi pressionado ou não, de modo que se esta sentença for verdadeira, mais uma condição é verificada (se a variável ***estadoSERVO*** é igual a 0), se esta for verdadeira, significa que o motor está parado, então o servo

é acionado através da função ***write()***, no entanto, se a mesma condição for falsa, sabemos que o motor está em movimento, logo ele é parado pela função ***stop()***.

Algo muito importante deve ser ressaltado neste momento. Perceba que a função ***write()*** utilizada dentro da função ***verificarBOT1()*** possui três parâmetros: o primeiro é determinado pela variável ***anguloFIN*** (ângulo desejado), o segundo é a velocidade com que o motor deve rotacionar (este valor varia entre 1 a 255, sendo que o valor 0 está atrelado à velocidade máxima), enquanto o terceiro argumento pode ser declarado como *TRUE* ou *FALSE*, se este for a primeira opção, o programa irá aguardar a conclusão do movimento, para então dar prosseguimento no código, porém se esta for *FALSE*, a execução do programa não será interrompida (por este motivo conseguimos parar o servo através do botão 1).

A função ***verificarBOT2()*** é responsável por estabelecer em qual linha do menu estará o cursor de seleção. Como dito anteriormente, pode-se escolher uma entre duas linhas para alteração das variáveis ***anguloFIN*** e ***velocidade***, desta forma a função em questão inverte o valor variável ***linhaATUAL*** (0 para a linha de ajuste de ***anguloFIN*** e 1 para a linha de ajuste da velocidade) e atualiza o valor da variável ***linhaANTERIOR***.

Já a função ***verificarBOT3()*** analisa se o botão 3 foi apertado, e caso tenha sido, é realizada uma análise posterior para saber em que linha está o cursor de seleção. Se o cursor de seleção citado estiver na linha referente à variável ***anguloFIN***, esta variável é incrementada em 10 unidades de forma que se este valor ultrapassar 180, a mesma variável recebe o valor 0, iniciando novamente o ciclo. Em contrapartida se o cursor de seleção estiver na linha referente à variável ***velocidade***, esta é incrementada em 5 unidades e caso atinja um valor maior do que 255, esta é zerada (Lembre-se o valor 0 nesta variável implica em velocidade máxima).

Por fim temos as funções ***verificarANGULO()*** e ***verificarLINHA()***. A primeira é responsável por fazer a leitura do ângulo atual do eixo do motor através da função ***read()*** e compará-lo ao valor existente na variável ***anguloANTERIOR*** para decidir se é necessário ou não atualizar o *display*. A segunda função serve unicamente para posicionar o cursor de seleção na linha correta.

```
void loop() {

  verificarBOT1();
  verificarBOT2();
  verificarBOT3();
```

```
  verificarANGULO();
  verificarLINHA();

}
```

```
void verificarBOT1(){

  leituraBOT1 = digitalRead(BOT1);

    if (leituraBOT1 !=  estadoantBOT1) {

        ultDebounceBOT1 = millis();

    }

    if ((millis() - ultDebounceBOT1) > tempoDeb) {

       if (leituraBOT1 != estadoatualBOT1) {

          estadoatualBOT1 = leituraBOT1;
         if (estadoatualBOT1 == 1) {

              if(estadoSERVO == 0){

                  servo.write(anguloFIN,velocidade,false);
                  estadoSERVO = 1;

              }
              else{

                 servo.stop();
                 estadoSERVO = 0;

              }
          }
       }
    }

    estadoantBOT1 = leituraBOT1;

}
```

```
void verificarBOT2(){

  leituraBOT2 = digitalRead(BOT2);

    if (leituraBOT2 !=  estadoantBOT2) {

        ultDebounceBOT2 = millis();

    }

    if ((millis() - ultDebounceBOT2) > tempoDeb) {

       if (leituraBOT2 != estadoatualBOT2) {

          estadoatualBOT2 = leituraBOT2;
         if (estadoatualBOT2 == 1) {

              linhaANTERIOR = linhaATUAL;
              linhaATUAL = !linhaATUAL;

           }
       }
    }

    estadoantBOT2 = leituraBOT2;

}
```

```
void verificarBOT3(){

  leituraBOT3 = digitalRead(BOT3);

    if (leituraBOT3 !=  estadoantBOT3) {

        ultDebounceBOT3 = millis();

    }

    if ((millis() - ultDebounceBOT3) > tempoDeb) {

       if (leituraBOT3 != estadoatualBOT3) {
```

```
  estadoatualBOT3 = leituraBOT3;

if (estadoatualBOT3 == 1) {

    if(linhaATUAL == 0)
    {

        lcd.setTextColor(WHITE);
        lcd.setCursor(0,27);
        lcd.print(">ANG FIN:");
        lcd.print(anguloFIN);

        anguloFIN = anguloFIN + 10;
        if(anguloFIN > 180){

            anguloFIN = 0;

        }

        lcd.setTextColor(BLACK);
        lcd.setCursor(0,27);
        lcd.print(">ANG FIN:");
        lcd.print(anguloFIN);
        lcd.display();

    }

    else{

         lcd.setTextColor(WHITE);
         lcd.setCursor(0,39);
         lcd.print(">VELOC:");
         lcd.print(velocidade);

        velocidade = velocidade + 5;
        if(velocidade > 255){

            velocidade = 0;

         }

         lcd.setTextColor(BLACK);
         lcd.setCursor(0,39);
```

```
                lcd.print(">VELOC:");
                lcd.print(velocidade);
                lcd.display();

            }
        }
    }
  }

  estadoantBOT3 = leituraBOT3;

}
```

```
void verificarLINHA(){

  if(linhaATUAL != linhaANTERIOR){

    if(linhaATUAL == 0){

      lcd.setTextColor(WHITE);
      lcd.setCursor(0,39);
      lcd.print(">");
      lcd.setTextColor(BLACK);
      lcd.setCursor(0,27);
      lcd.print(">");
      lcd.display();

    }
    else{

      lcd.setTextColor(WHITE);
      lcd.setCursor(0,27);
      lcd.print(">");
      lcd.setTextColor(BLACK);
      lcd.setCursor(0,39);
      lcd.print(">");
      lcd.display();

    }

  }
```

```
}

void verificarANGULO(){

    anguloATUAL = servo.read();

    if(anguloATUAL != anguloANTERIOR){

        lcd.setTextColor(WHITE);
        lcd.setCursor(0,15);
        lcd.print(" ANG AT:");
        lcd.print(anguloANTERIOR);
        lcd.setTextColor(BLACK);
        lcd.setCursor(0,15);
        lcd.print(" ANG AT:");
        lcd.print(anguloATUAL);
        lcd.display();

        anguloANTERIOR = anguloATUAL;
    }
}
```

APÊNDICE A – Códigos

Este apêndice contém todos os códigos de desenvolvidos nos itens referentes à este documento, no entanto aqui estes não se encontram de forma segmentada (maneira que foram utilizados nos itens já citados).

OBS: Para conferir se os códigos estavam corretos resolvi por várias vezes copia-los diretamente deste documento e colar os mesmos na IDE do Arduino para compilar e assim testar, no entanto, existe a possibilidade de esta acusar alguns erros na compilação devido ao fato de alguns caracteres serem passados de maneira errada, como é o caso do "-" (sinal de menos), portanto caso isso aconteça, basta ir ao local onde está escrito este caractere e troca-lo pelo mesmo porém escrito diretamente na IDE.

CÓDIGO DO ITEM 1

```
#include <SPI.h>
#include <Adafruit_PCD8544.h>
#include <Adafruit_GFX.h>

Adafruit_PCD8544 lcd = Adafruit_PCD8544(7, 6, 5, 4, 3);

void setup() {

    lcd.begin();
    delay(2000);

    lcd.clearDisplay();

    lcd.setTextSize(1);
    lcd.setTextColor(BLACK);
    lcd.setCursor(22,20);
    lcd.print("EXEMPLO");

    lcd.display()
 }
 void loop(){
 }
```

CÓDIGO DO ITEM 2

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>
#include <Servo.h>

Adafruit_PCD8544 lcd = Adafruit_PCD8544(7, 6, 5, 4, 3);
Servo servo;

int angulo= 27;

void setup() {

    servo.attach(10);

    lcd.begin();
    delay(2000);
    lcd.clearDisplay();

    lcd.setTextSize(1);
    lcd.setTextColor(BLACK);
    lcd.setCursor(25,20);
    lcd.print("ANG:");
    lcd.print(angulo);

    lcd.display();

    servo.write(angulo);

}

void loop() {
```

```
}
```

CÓDIGO DO ITEM 3

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>
#include <Servo.h>

#define POT1 0
#define POT2 1
#define POT3 2
#define POT4 3

Adafruit_PCD8544 lcd = Adafruit_PCD8544(7, 6, 5, 4, 3);

Servo servo1;
Servo servo2;
Servo servo3;
Servo servo4;

int medidaPOT1;
int medidaPOT1ant;

int medidaPOT2;
int medidaPOT2ant;

int medidaPOT3;
int medidaPOT3ant;
int medidaPOT4;
int medidaPOT4ant;

void setup() {
```

```
  iniciarSERVOS();
  iniciarLCD();

}

void loop() {

 atualizarSERVOS();

}

void iniciarLCD(){

    lcd.begin();
    delay(2000);
    lcd.clearDisplay();

    lcd.drawRect(0,0,41, 23, BLACK);
    lcd.drawRect(42,0,42, 23, BLACK);
    lcd.drawRect(0,24,41, 23, BLACK);
    lcd.drawRect(42,24,42, 23, BLACK);

    lcd.setTextSize(1);
    lcd.setTextColor(BLACK);
    lcd.setCursor(15,2);
    lcd.print("S1");

    lcd.setCursor(57,2);
    lcd.print("S2");

    lcd.setCursor(15,26);
    lcd.print("S3");

    lcd.setCursor(57,26);
    lcd.print("S4");
```

```
    lcd.setCursor(2,13);
    lcd.print(" A:");
    lcd.print(medidaPOT1);

    lcd.setCursor(44,13);
    lcd.print(" A:");
    lcd.print(medidaPOT2);

    lcd.setCursor(2,37);
    lcd.print(" A:");
    lcd.print(medidaPOT3);

    lcd.setCursor(44,37);
    lcd.print(" A:");
    lcd.print(medidaPOT4);

    lcd.display();

}

void iniciarSERVOS(){

    servo1.attach(8);
    servo2.attach(9);
    servo3.attach(10);
    servo4.attach(11);


    medidaPOT1 = analogRead(POT1);
    medidaPOT1 = map(medidaPOT1,0,1023,0,180);

    medidaPOT2 = analogRead(POT2);
    medidaPOT2 = map(medidaPOT2,0,1023,0,180);

    medidaPOT3 = analogRead(POT3);
```

```
    medidaPOT3 = map(medidaPOT3,0,1023,0,180);

    medidaPOT4 = analogRead(POT4);
    medidaPOT4 = map(medidaPOT4,0,1023,0,180);

    servo1.write(medidaPOT1);
    servo2.write(medidaPOT2);
    servo3.write(medidaPOT3);
    servo4.write(medidaPOT4);

    medidaPOT1ant = medidaPOT1;
    medidaPOT2ant = medidaPOT2;
    medidaPOT3ant = medidaPOT3;
    medidaPOT4ant = medidaPOT4;

}

void atualizarSERVOS(){

    medidaPOT1 = analogRead(POT1);
    medidaPOT1 = map(medidaPOT1,0,1023,0,180);

     if (medidaPOT1 != medidaPOT1ant){

          lcd.setTextSize(1);
          lcd.setCursor(2,13);
          lcd.print(" A:");
          lcd.setTextColor(WHITE);
          lcd.print(medidaPOT1ant);
          lcd.setTextColor(BLACK);
          lcd.setCursor(2,13);
          lcd.print(" A:");
          lcd.print(medidaPOT1);
          lcd.display();
```

```
        medidaPOT1ant = medidaPOT1;
        servo1.write(medidaPOT1);
    }

    medidaPOT2 = analogRead(POT2);
    medidaPOT2 = map(medidaPOT2,0,1023,0,180);

    if (medidaPOT2 != medidaPOT2ant){

        lcd.setTextSize(1);
        lcd.setCursor(44,13);
        lcd.print(" A:");
        lcd.setTextColor(WHITE);
        lcd.print(medidaPOT2ant);
        lcd.setTextColor(BLACK);
        lcd.setCursor(44,13);
        lcd.print(" A:");
        lcd.print(medidaPOT2);
        lcd.display();

        medidaPOT2ant = medidaPOT2;
        servo2.write(medidaPOT2);
    }

    medidaPOT3 = analogRead(POT3);
    medidaPOT3 = map(medidaPOT3,0,1023,0,180);

    if (medidaPOT3 != medidaPOT3ant){

        lcd.setTextSize(1);
        lcd.setCursor(2,37);
        lcd.print(" A:");
        lcd.setTextColor(WHITE);
        lcd.print(medidaPOT3ant);
        lcd.setTextColor(BLACK);
```

```
        lcd.setCursor(2,37);
        lcd.print(" A:");
        lcd.print(medidaPOT3);
        lcd.display();

        medidaPOT3ant = medidaPOT3;
        servo3.write(medidaPOT3);
    }

    medidaPOT4 = analogRead(POT4);
    medidaPOT4 = map(medidaPOT4,0,1023,0,180);

    if (medidaPOT4 != medidaPOT4ant){

        lcd.setTextSize(1);
        lcd.setCursor(44,37);
        lcd.print(" A:");
        lcd.setTextColor(WHITE);
        lcd.print(medidaPOT4ant);
        lcd.setTextColor(BLACK);
        lcd.setCursor(44,37);
        lcd.print(" A:");
        lcd.print(medidaPOT4);
        lcd.display();

        medidaPOT4ant = medidaPOT4;
        servo4.write(medidaPOT4);
   }
}
```

CÓDIGO DO ITEM 4

```
#include <SPI.h>
#include <Adafruit_GFX.h>
```

```
#include <Adafruit_PCD8544.h>
#include <VarSpeedServo.h>

#define BOT1 0
#define BOT2 1
#define BOT3 2

Adafruit_PCD8544 lcd = Adafruit_PCD8544(7, 6, 5, 4, 3);
VarSpeedServo servo;

bool estadoSERVO = 0;
int velocidade = 0;

int anguloANTERIOR=0;
int anguloATUAL=0;
int anguloFIN = 0;

bool estadoatualBOT1 = 0;
bool estadoantBOT1 = 0;
bool leituraBOT1 = 0;
long ultDebounceBOT1 = 0;


bool estadoatualBOT2 = 0;
bool estadoantBOT2 = 0;
bool leituraBOT2 = 0;
long ultDebounceBOT2 = 0;

bool estadoatualBOT3 = 0;
bool estadoantBOT3 = 0;
bool leituraBOT3 = 0;
long ultDebounceBOT3 = 0;

long tempoDeb = 50;


bool linhaATUAL;
bool linhaANTERIOR;


void setup() {

    pinMode(BOT1,INPUT);
```

```
    pinMode(BOT2,INPUT);
    pinMode(BOT3,INPUT);

    servo.attach(8);
    servo.write(0);

    lcd.begin();
    delay(2000);
    lcd.clearDisplay();

    lcd.setTextSize(1);
    lcd.setTextColor(BLACK);
    lcd.setCursor(27,0);
    lcd.print("SERVO");
    lcd.drawLine(0, 10, 84, 10, BLACK);

    lcd.setCursor(0,15);
    lcd.print(" ANG AT:0");

    lcd.setCursor(0,27);
    lcd.print(">ANG FIN:0");

    lcd.setCursor(0,39);
    lcd.print(" VELOC:0");

    lcd.display();

}


void loop() {

  verificarBOT1();
  verificarBOT2();
  verificarBOT3();
  verificarANGULO();
  verificarLINHA();

}


void verificarBOT1(){

  leituraBOT1 = digitalRead(BOT1);
```

```
    if (leituraBOT1 !=  estadoantBOT1) {

        ultDebounceBOT1 = millis();

    }

    if ((millis() - ultDebounceBOT1) > tempoDeb) {

       if (leituraBOT1 != estadoatualBOT1) {

          estadoatualBOT1 = leituraBOT1;
         if (estadoatualBOT1 == 1) {

              if(estadoSERVO == 0){

                  servo.write(anguloFIN,velocidade,false);
                  estadoSERVO = 1;

              }
              else{

                 servo.stop();
                 estadoSERVO = 0;

              }
           }
       }
    }

    estadoantBOT1 = leituraBOT1;

}


void verificarBOT2(){

  leituraBOT2 = digitalRead(BOT2);

    if (leituraBOT2 !=  estadoantBOT2) {

        ultDebounceBOT2 = millis();

    }
```

```
    if ((millis() - ultDebounceBOT2) > tempoDeb) {

        if (leituraBOT2 != estadoatualBOT2) {

            estadoatualBOT2 = leituraBOT2;
          if (estadoatualBOT2 == 1) {

              linhaANTERIOR = linhaATUAL;
              linhaATUAL = !linhaATUAL;

            }
        }
    }

    estadoantBOT2 = leituraBOT2;

}

void verificarBOT3(){

  leituraBOT3 = digitalRead(BOT3);

    if (leituraBOT3 !=  estadoantBOT3) {

        ultDebounceBOT3 = millis();

    }

    if ((millis() - ultDebounceBOT3) > tempoDeb) {

        if (leituraBOT3 != estadoatualBOT3) {

            estadoatualBOT3 = leituraBOT3;

          if (estadoatualBOT3 == 1) {

              if(linhaATUAL == 0)
              {

                  lcd.setTextColor(WHITE);
                  lcd.setCursor(0,27);
                  lcd.print(">ANG FIN:");
                  lcd.print(anguloFIN);
```

```
                anguloFIN = anguloFIN + 10;
                if(anguloFIN > 180){

                    anguloFIN = 0;

                }

                lcd.setTextColor(BLACK);
                lcd.setCursor(0,27);
                lcd.print(">ANG FIN:");
                lcd.print(anguloFIN);
                lcd.display();

            }

            else{

                lcd.setTextColor(WHITE);
                lcd.setCursor(0,39);
                lcd.print(">VELOC:");
                lcd.print(velocidade);

                velocidade = velocidade + 5;
                if(velocidade > 255){

                    velocidade = 0;

                }

                lcd.setTextColor(BLACK);
                lcd.setCursor(0,39);
                lcd.print(">VELOC:");
                lcd.print(velocidade);
                lcd.display();

            }
        }
    }
  }

  estadoantBOT3 = leituraBOT3;

}
```

```
void verificarLINHA(){

   if(linhaATUAL != linhaANTERIOR){

     if(linhaATUAL == 0){

        lcd.setTextColor(WHITE);
        lcd.setCursor(0,39);
        lcd.print(">");
        lcd.setTextColor(BLACK);
        lcd.setCursor(0,27);
        lcd.print(">");
        lcd.display();

     }
     else{

        lcd.setTextColor(WHITE);
        lcd.setCursor(0,27);
        lcd.print(">");
        lcd.setTextColor(BLACK);
        lcd.setCursor(0,39);
        lcd.print(">");
        lcd.display();

     }

  }

}


void verificarANGULO(){

    anguloATUAL = servo.read();

    if(anguloATUAL != anguloANTERIOR){

          lcd.setTextColor(WHITE);
          lcd.setCursor(0,15);
          lcd.print(" ANG AT:");
          lcd.print(anguloANTERIOR);
          lcd.setTextColor(BLACK);
          lcd.setCursor(0,15);
```

```
            lcd.print(" ANG AT:");
            lcd.print(anguloATUAL);
            lcd.display();

            anguloANTERIOR = anguloATUAL;
        }
}
```