ROPIC

Controle de servo-motores através do PIC18F4550

ANDRÉ MALTA
CEFET-MG CAMPUS TIMÓTEO

1. SERVO-MOTOR

O servo-motor é um dispositivo eletromecânico, que apresenta movimento proporcional a um comando, como dispositivos de malha fechada, ou seja, recebem um sinal de controle, verifica a posição atual para controlar o seu movimento e vai para a posição desejada. Para isso, ele é composto por algumas partes, que estão descritas abaixo:

- Circuito de Controle: responsável pelo monitoramento do potenciômetro e acionamento do motor visando obter uma posição pré-determinada;
- Potenciômetro: ligado ao eixo de saído do servo, monitora a posição do mesmo;
- Motor: movimenta as engrenagens e o eixo principal do servo;
- Engrenagens: reduzem a rotação do motor, transferem mais toque ao eixo principal de saída e movimenta o potenciômetro junto com o eixo;
- Caixa do Servo: caixa para acondicionar as diversas partes do servo.

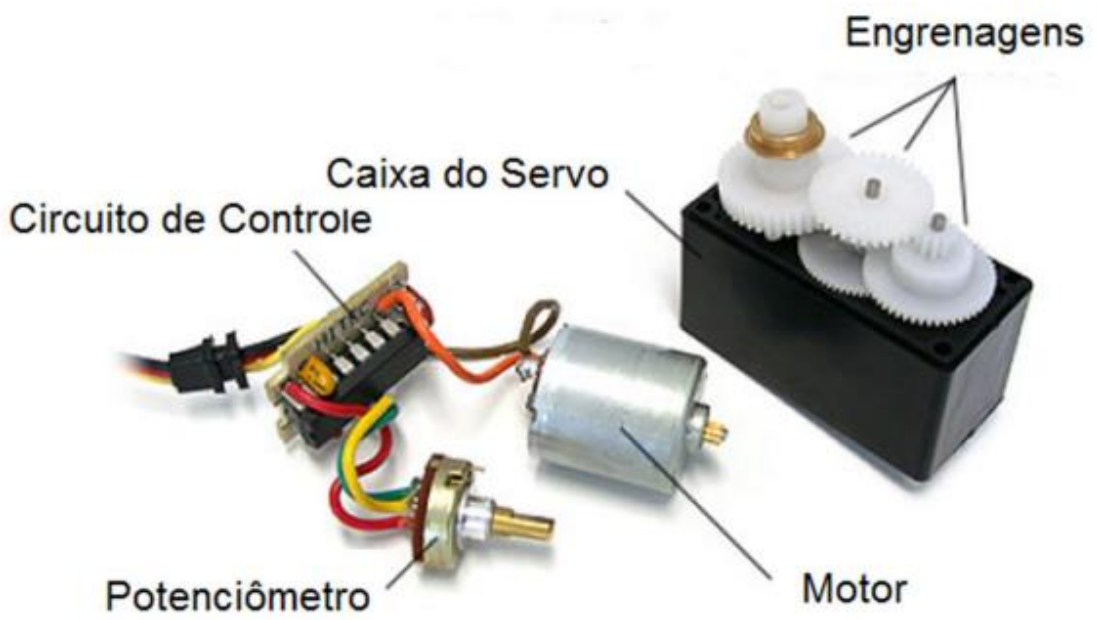

Figura 1 - Componentes internos do servo-motor

2. CONTROLE DO SERVO-MOTOR

O servo-motor é alimentado com tensões de 5v e recebe o sinal no formato PWM (Pulse Width Modulation), variando de 0v ou 5v. Para controle é necessário realizar um pulso a cada 20ms, ou seja, frequência de 50Hz. O comprimento do pulso determina a distância que o motor gira, um sinal com largura de pulso de 1ms corresponde a posição do servo todo a esquerda ou a 0 grau; um sinal com largura de pulso de 1,5ms corresponde a posição central do servo ou de 90 graus; um sinal com largura de pulso de 2ms corresponde a posição do servo todo a direita ou 180 graus. Podendo ser usado qualquer valor intermediário entre 1ms e 2ms para posições proporcionalmente intermediarias.

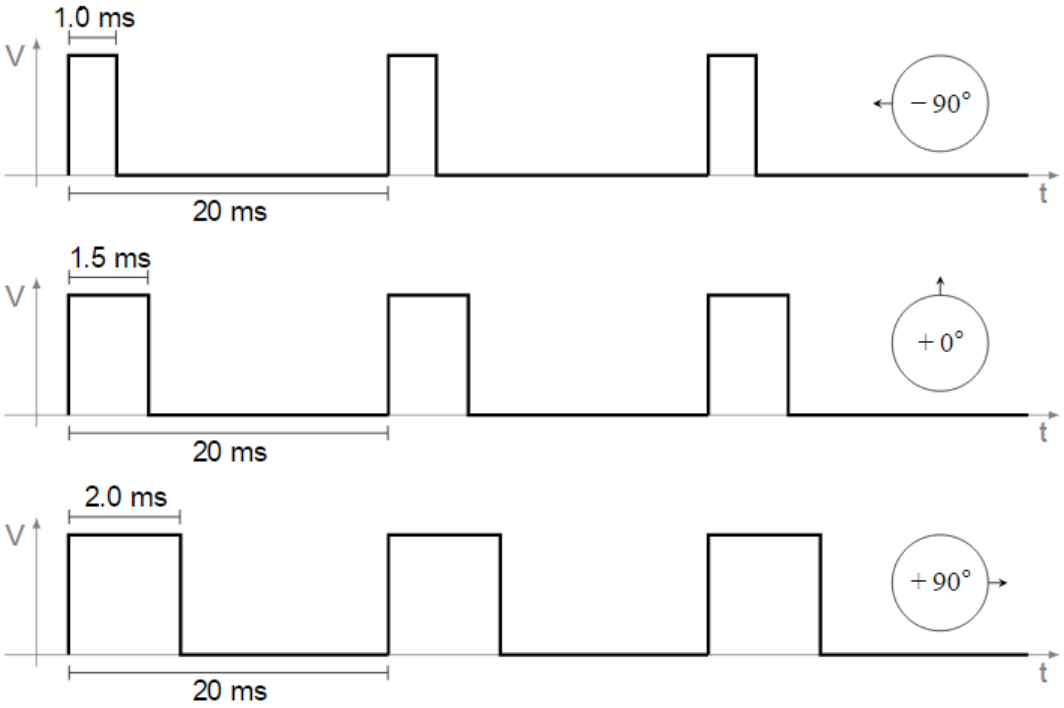

Figura 2 - Controle PWM

3. SERVO-MOTOR COM PIC

Para a realização do controle dos servo-motores o PIC18F4550 foi utilizado, juntamente com um Crystal externo, que pode ser de 16MHz ou 20MHz, para cada um é carregado a configuração para base de cálculos de acordo com a biblioteca desenvolvida, visando

obter melhora na precisão. Contudo, para melhor exemplificar a lógica aplicada para o funcionamento do controle dos servos-motores, vamos utilizar o clock de 16Mhz.

O PIC18F4550 consta com vários módulos de Timer, sendo utilizado para o projeto o Timer 1 configurado com 16bits e prescaler 1:4.

Como descrito no tópico anterior, o servo-motor possui um período completo de 20ms, então para utilizar até 8 servos, de maneira multiplexada, precisamos de um tempo de 20/8 = 2,5ms, este tempo é o subperíodo definido para cada um dos servo-motores. Dentro desse tempo, é realizado o controle de cada servo, um após o outro. Podemos reparar isso na imagem abaixo, que mostra um exemplo de controle multiplexado dos servos.

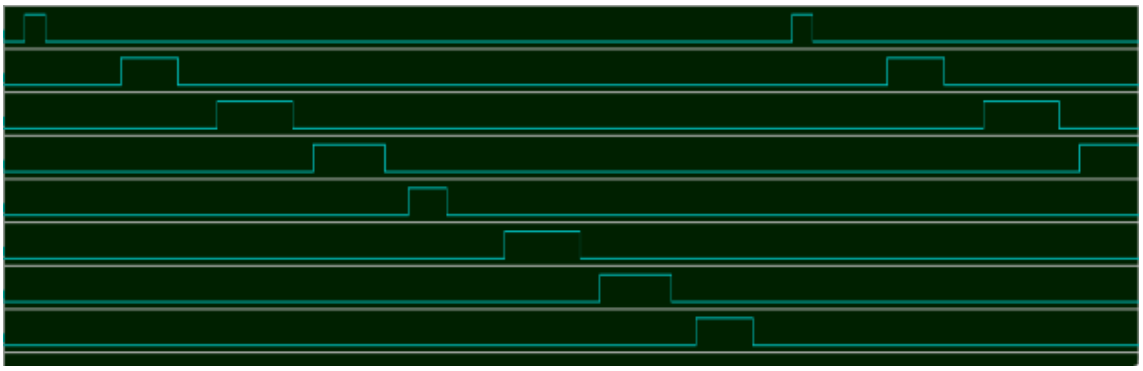

Figura 3 - Servos multiplexados

Esta configuração no Timer 1 gera um pulso de contagem de 1us, contudo se, queremos um tempo de 1,5ms (nível alto) simplesmente faremos o timer1 contar até 1500, ou seja, carregar o timer1 com 65535-1500 = 64035. O tempo restante do subperíodo definido para o servo-motor de 2,5ms deve ser em nível baixo, então devemos carregar novamente o timer1 com 65535-1000 = 64535, gerando um estouro de 1ms para completar o subperíodo.

Basicamente é definido o nível logico alto para o pino, carregado o timer 1 com valor respectivo a quantidade de graus que se deseja movimentar (entre 1000 e 2000, como visto no tópico anterior) e espera o estouro do timer 1. Logo é definido o nível logico baixo para o pino, carregado então o timer 1 com o tempo restante do subperíodo, espera o estouro e repete o ciclo para um outro servo. Os cálculos para um clock de 20MHz são os mesmos, somente é realizado em outra escala, a mudança será no pulso que passa a ser de 0.8us, diferentemente do exemplificado que é de 1us.

4. SERVO HS-422

Servo-motor fabricado pela empresa Hitec, de alta qualidade e simples para controle, abaixo está listado algumas especificações importantes:

- Tensão de Funcionamento: 4,8-6,0Volts;
- Faixa de Temperatura: -20 a +60 graus C;
- Peso: 45,5g;
- Dimensões: 40,6x19,8x36,6mm;
- Ângulo de Funcionamento: 180 graus;
- Consumo de Corrente: (4,8V) 8mA/Ocioso – (6,0V) 8,8mA/Ocioso;
- Torque: (4.8V) 3,3kg.cm - (6,0V) 4,1kg.cm;
- Frequência: 50Hz;
- Largura de Pulso: 600us (0 graus) a 2400us (180 graus).

Figura 4 - Servo-motor Hs-422

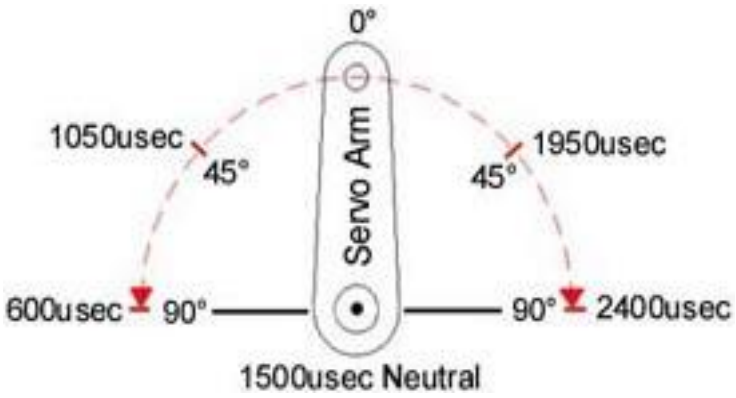

Figura 5 – Largura de Pulso

5. DESENHO DO CIRCUITO

Para realização do projeto de controle dos servo-motores, foram utilizados os seguintes itens:

- 2 capacitores de 22pF;
- 1 crystal de 20MHz;

- 8 servo-motores HS-422;
- 1 PIC18F4550.
- 1 display de 7 segmentos catodo comum
- 7 resistores de 330 ohms
- 4 botões
- 4 resistores de 10k ohms

Abaixo segue o desenho do circuito de controle elaborado:

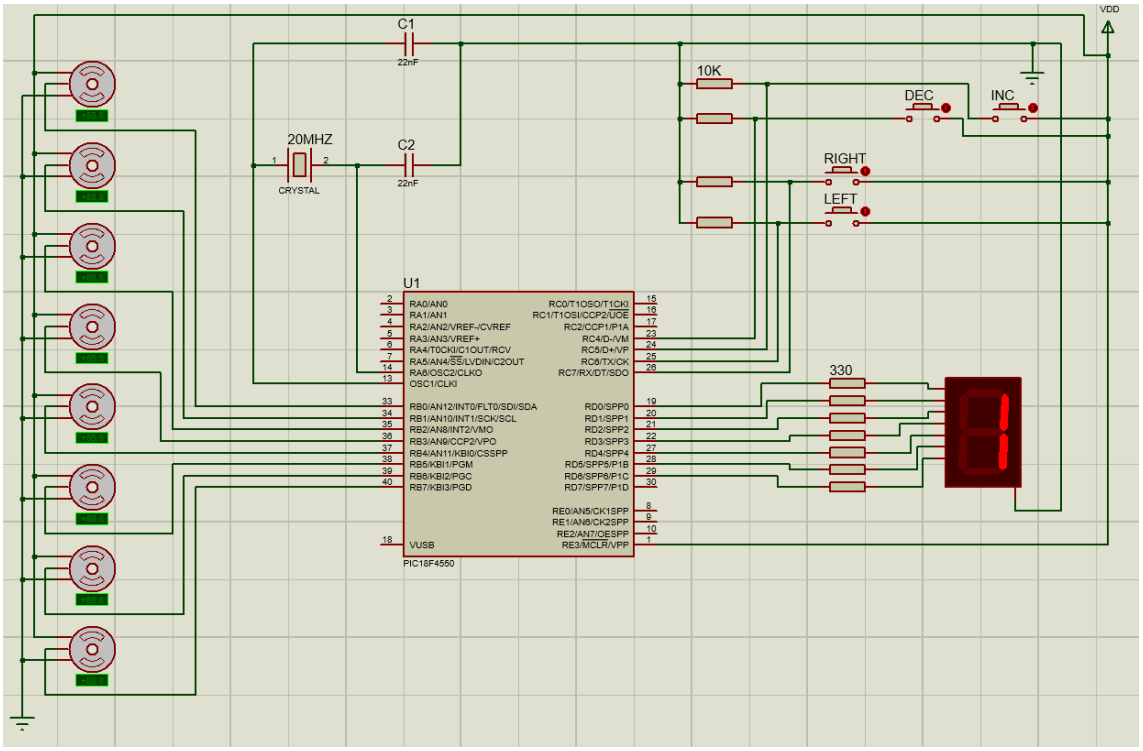

Figura 6 - Desenho do circuito para controle dos servo-motores

O projeto tem como objetivo demonstrar o uso da biblioteca desenvolvida, dessa maneira foi adicionado 2 botões para selecionar o motor desejado, que é mostrado no display. Feito a seleção os botões RIGHT e LEFT realizam o movimento do motor selecionado.

6. BIBLIOTECA SERVO.H

A biblioteca “servo.h” foi desenvolvida no intuito de facilitar o controle de cada servo, de maneira multiplexada, seguindo a lógica de interrupções do Timer1 descrita e exemplificada no tópico anterior, segue abaixo suas especificações, restrições e descrições de suas funcionalidades:

6.1 FUNÇÕES DO SERVO

Funções	Descrição
SetClockBase	Seleciona o clock para realização da base de cálculos do Timer1, somente é possível configura-lo para 16MHz ou 20MHz. Como padrão clocl de 20MHZ.
SetPulseWidth	Define a largura do pulso utilizado pelo servo-motor. Obs: cada fabricante possui suas especificações.
SetServoScale	Configura a posição mínima e máxima dos servo-motores. Obs: somente permitido valores maiores que 0.
InitServos	Prepara para uso os servo-motores e configura a porta de saída do pulso, sendo PORTB ou PORTD.
MoveServoAsync	Movimenta o servo-motor desejado para a posição informada de forma assíncrona.
MoveAllServos	Movimenta todos os servo-motores de maneira assíncrona com a mesma posição informada.
GetServoPosition	Retorna a posição atual do servo-motor.
GetMinPosition	Retorna a posição mínima.
GetMaxPosition	Retorna a posição máxima.

Tabela 1 - Funções do servo-motor

6.2 DESCRIÇÃO DAS FUNÇÕES

SetClockBase

Função: Configura a base de cálculo do timer1 de acordo com o clock.

Protótipo: *void SetClockBase(double CLOCK_BASE);*

Argumentos: *CLOCK_BASE*

CLOCK_16MHz: Definição da Base de cálculo de 1us de pulso para realização dos cálculos de estouro do Timer1 de acordo com a posição do servo informada;

CLOCK_20MHz: Definição da Base de cálculo de 0.8us de pulso para realização dos cálculos de estouro do Timer1 de acordo com a posição do servo informada.

Observações: Por padrão a base de calculo já está definida para trabalhar com o clock de 20MHz, então, só se faz necessário a utilização desta função caso o clock seja diferente, ou seja, de 16MHz;

Outros valores diferentes de 16Mhz e 20MHz não estão configurados, logo não é possível a utilização sem haver modificações na biblioteca.

Exemplo: *SetClockBase(CLOCK_16MHz);* ou
SetClockBase(CLOCK_20MHz);

SetPulseWidth

Função: Configura a largura do pulso para controle dos servo-motores.

Protótipo: *void SetPulseWidth(int minPulse_us, int maxPulse-us);*

Argumentos: *minPulse_us, maxPulse_us*

Observações: Cada modelo de servo-motor possui suas especificações para controle, logo esta função deverá ser utilizada se, somente se houver alguma alteração no valor do pulso mínimo e máximo do servo-motor utilizado;

Os pulsos minimo de 1000(1ms) e 2000(2ms) estão definidos como padrão da biblioteca, então não é necessário o uso da função quando o servo possuir estas configurações;

Os valores informados são em us.

Exemplo: *SetPulseWidth(0,180);*

SetServoScale

Função: Configura a escala de posição, mínima e máxima para controle dos servo-motores.

Protótipo: *void SetServoScale(int min, int max);*

Argumentos: *min, max*

Observações: Somente é possível passar valores maiores que 0 para esta função, estes números são definidos para funcionarem como uma escala de referência para controle dos servo-motores;

Feito a configuração dos graus, os valores informados para movimentação do servo-motor que não estejam dentro da escala definida, não será atribuído ao mesmo, consequentemente o servo permanecera na posição inicial igual a 0 grau;

Se a função não for utilizada, por padrão, a escala passa a ser por padrão a definida na largura de pulso através da função *SetPulseWidth,* caso não definida, passa a ser de 1000(posição inicial igual a 0 grau), referente ao

pulso de 1ms, até 2000(posição final igual a 180 graus), referente ao pulso de 2ms.

Exemplo: *SetServoScale(0,180);*

InitServos

Função: Prepara os servo-motores para uso.

Protótipo: *void InitServos(unsigned short nServos, char port);*

Argumentos: *nServos, port*

***nServos*:** Quantidade de servo-motores a serem utilizados, sendo máximo permitido igual a 8;

port: O endereço da porta de saída de controle do PIC, somente é possível fazer uso do PORTB ou PORTD.

Observações: Inicializa a contagem do Timer1 de acordo com o clock base(padrão 20MHz), prepara os servos para uso, mas não os habilita, portanto não inicia o movimento.

Exemplo: *InitServos(8, (char) &PORTB);*

MoveServoAsync

Função: Habilita e movimenta de forma assíncrona o servo-motor especificado.

Protótipo: *void MoveServoAsync(char servo_number, int degree);*

Argumentos: *servo_number, degree*

***servo_number*:** Número do servo-motor para realização do movimento assíncrono.

degree: Posição absoluta correspondente para onde o servo-motor deve ir.

Observações: O número do servo especificado para controle deve estar entre a quantidade de servo-motores preparados pela função *InitServos,* caso sejam preparados 8 servos, o argumento servo_number deve estar entre 1 e 8;

A posição informada deve estar entre a escala definida pela função *SetServoScale* se utilizada;

Como a movimentação acontece de forma assíncrona, todos os motores movimentam juntos.

Exemplo:
MoveServoAsync(1, 1072);
MoveServoAsync(2, 1500);
MoveServoAsync(3, 1200);

MoveServoAsync(4, 1000);
MoveServoAsync(5, 1784);
MoveServoAsync(6, 1323);
MoveServoAsync(7, 1900);
MoveServoAsync(8, 2000);

MoveAllServos

Função: Habilita e movimenta de forma assíncrona todos os servo-motores inicializados.

Protótipo: *void MoveAllServos(int degree);*

Argumentos: *degree*

degree: Posição absoluta correspondente para onde o servo-motor deve ir.

Observações: A posição informada deve estar entre a escala definida pela função *SetServoScale* se utilizada;

Como a movimentação acontece de forma assíncrona, todos os motores se movimentam de maneira multiplexada.

Exemplo: *MoveAllServos(2000);*

GetServoPosition

Função: Retorna a posição atual do servo-motor selecionado.

Protótipo: *int GetServoPosition(int servo_number);*

Argumentos: *servo_number*

***servo_number*:** Número do servo-motor para retorno de sua posição atual.

Exemplo: int pos = *GetServoPosition(2);*

GetMinPosition

Função: Retorna a posição mínima base do movimento.

Protótipo: *int GetMinPosition();*

Exemplo: int minPos = *GetMinPosition();*

GetMaxPosition

Função: Retorna a posição máxima base do movimento.

Protótipo: *int GetMaxPosition();*

Exemplo: int maxPos = *GetMaxPosition();*

6.3 EXEMPLO DE USO

```
#include <p18f4550.h>
#include “servos.h”
#include <delays.h>

void main(void){
        TRISB = 0;
        SetClockBase(CLOCK_20MHz);
        SetPulseWidth(600, 2400);
        SetServoScale(0, 180);
        InitServos(8, (char) &PORTB);
        //Movimento 1
        MoveServoAsync(1, 13);
        MoveServoAsync(2, 90);
        MoveServoAsync(3, 36);
        MoveServoAsync(4, 0);
        MoveServoAsync(5, 141);
        MoveServoAsync(6, 58);
        MoveServoAsync(7, 162);
        MoveServoAsync(8, 180);
        Delay10KTCYx(250);
        Delay10KTCYx(250);
        //Movimento 2
        MoveAllServos(180);
        while (1);
}
```

6.4 RESULTADO DA EXECUÇÃO

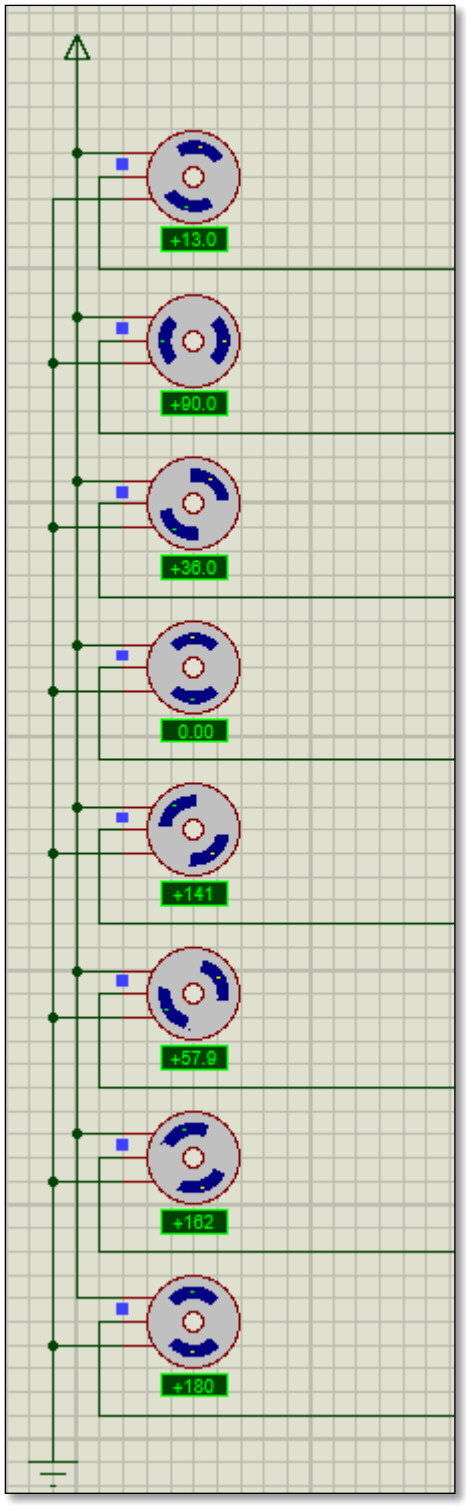

Figura 7 - Movimento 1

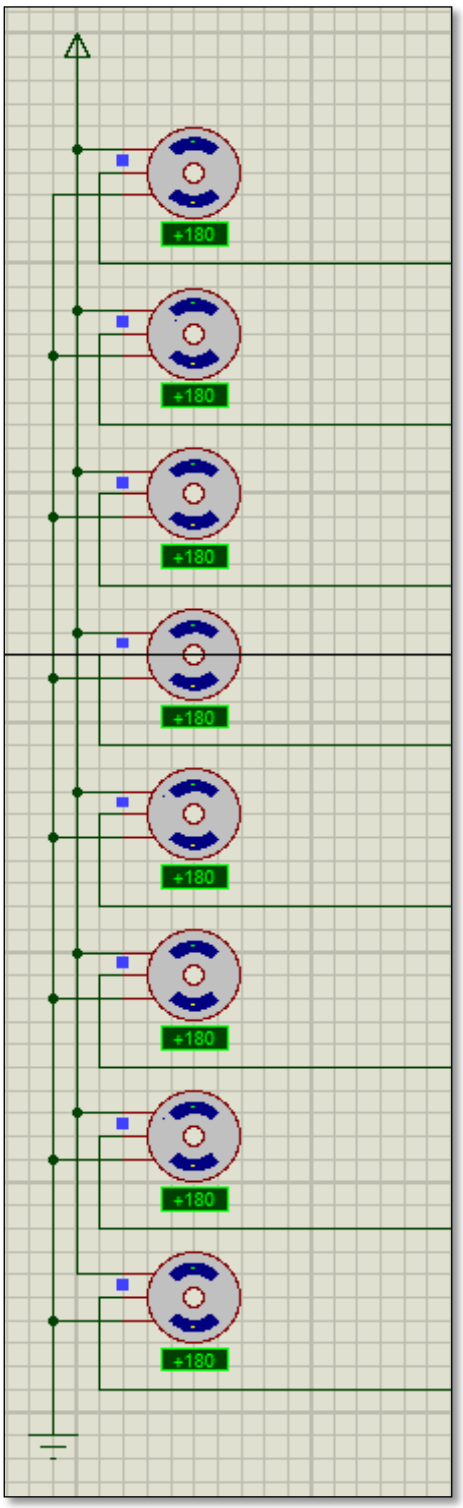

Figura 8 – Movimento 2

7. REFERÊNCIAS

Servo Motor, UNESP – Departamento de Engenharia Elétrica. Disponível em: < http://www.feis.unesp.br/Home/departamentos/engenhariaeletrica/aula-4---servo-motor-13-03-2013-final.pdf> Acesso em 25 de junho de 2016.

HS-422 Super Soport, Servocity. Disponível em: < https://www.servocity.com/html/hs-422_super_sport_.html#.V3R1l7grKUl> Acesso em 28 de junho de 2016.

Microship Technology Inc. Datasheet: PIC18F2455/2550/4455/4550, 28/40/44-Pin, Hight-Performancem Enhanced Flash, USB Microcontrollers with nano Watt Technology. 2006.