Prótese do membro superior direito (mão)

Proposta do Projeto

Júlio Ohfugi Yamaguti
FGA
UNB – Universidade de Brasília
Gama,Brasil
julio.ohfugi21@gmail.com

Tiago Ribeiro Freire
FGA
UNB – Universidade de Brasília
Gama,Brasil
tiagorff2@gmail.com

***Resumo*- Este documento demonstra de forma detalhada a composição de hardware e software para a produção de uma prótese de mão.**

I. INTRODUÇÃO

Nos últimos anos a tecnologia tem caminhado para solucionar muitos problemas no âmbito das limitações físicas, como deficiências na locomoção, na fala, na audição e etc. E empresas que investiram seus esforços e investimentos nessa área tiveram e têm uma boa lucratividade, como a RSL Steeper. São empresas que têm que estar aplicando melhorias constantemente em suas tecnologias, pois são exigidos movimentos muito delicados e precisos.

Em diversos casos de acidentes graves, com a necessidade de amputação, ou mesmo de problemas de nascença com algum tipo de malformação de membro, há o grande problema em que o indivíduo possui a ausência da mão e, consequentemente, a perda do papel fundamental que essa parte do corpo tão importante tem. Como por exemplo movimentos simples de abrir e fechar ou até mesmo de segurar uma sacola qualquer.

Com isso, surge a ideia de um projeto com o intuito de suprir essa necessidade e trazer de volta, mesmo que parcialmente, a mobilidade que essa parte do corpo proporciona. Sendo assim, optou-se por desenvolver um protótipo em que a partir de uma sequência de contrações do músculo tríceps braquial o sensor piezoeléctrico irá captar os pulsos e através de um código compilado no microcontrolador MSP430 os motores servos irão trabalhar em determinada angulação para fazer com que os movimentos da mão artificial seja semelhantes aos movimentos de uma mão normal e supra as necessidades do dia a dia. Assim, trazendo de uma vez por todas a confiança e a auto-estima de quem passou anos com essa debilidade.

II. OBJETIVOS

Este projeto tem como objetivo a criação de um protótipo que simula uma mão através de uma prótese artificial, com o intuito de trazer de volta movimentos básicos ao indivíduo que necessite. Além disso, tem intenção de ser algo com preço em conta e, sendo assim, acessível a todas as classes da sociedade.

III. REQUISITOS

A prótese proposta deve possuir um controle que possibilite que o usuário consiga manuseá-la de forma dinâmica e clara. Além disso, deve ter a força necessária para segurar e manipular objetos, e seu consumo energético deve ser baixo e econômico para que a bateria tenha uma duração considerável.

IV. Benefícios

A recuperação de movimentos básicos facilitadores dos afazeres do dia-a-dia, a recuperação da auto-estima pessoal do usuário e a autonomia em diversas situações que, sem a prótese, haveria uma certa dependência.

V. Materiais Utilizados

- 2 Servomotores TowerPro 9g SG-90;
- MSP430G2553;
- Jumpers macho e fêmea;
- Mangueira;
- 1 Piezoelétrico;
- Linha de nylon;
- Fita isolante
- Superfície de plástico;
- Elástico

VI. Descrição de hardware

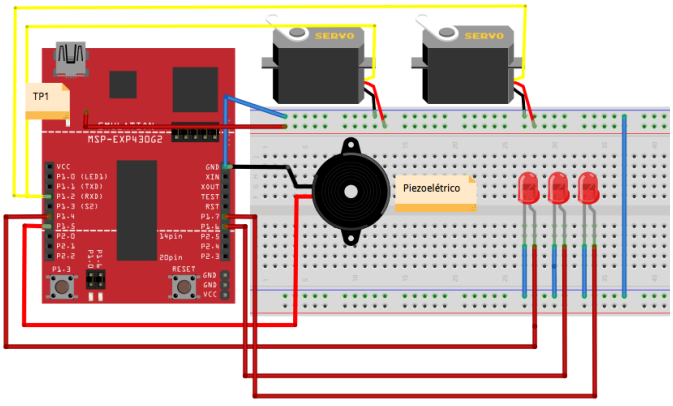
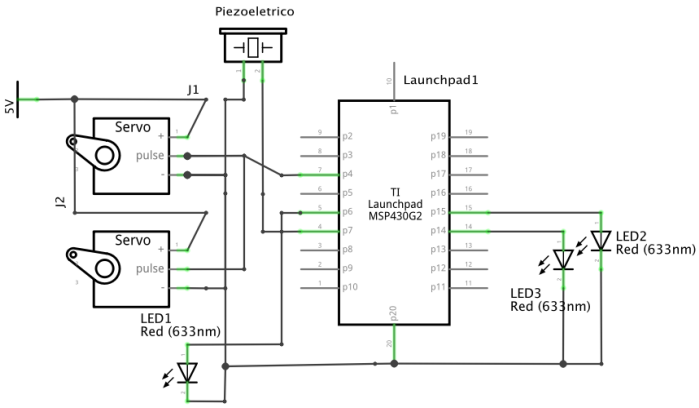


Figura 1 – Esquemático de ligação dos componentes eletrônicos com o MSP430.

Soldou-se um pino na saída TP1 da placa MSP430 para obter-se 5V de alimentação, proveniente do cabo USP que liga o MSP, com o intuito de alimentar os servomotores que operam com esse nével de tensão. Além disso, conectou-se o piezoelétrico ao ground e ao pino P1.3 e as saídas dos servos aos pinos P1.1 e P1.2 e conectou-se os leds aos pinos 4, 6, 7 e 8.



Figuara 2 - Esquemático das ligações.

VII. Descrição de software

A linguagem utilizada para programação foi C, e seguiu a lógica representada pelo diagrama de blocos a seguir:

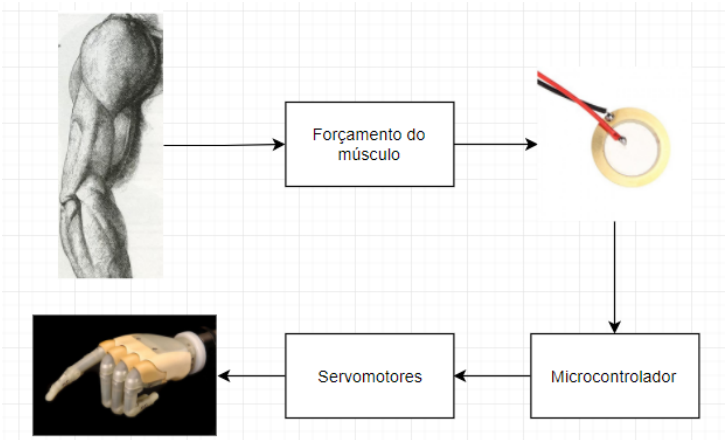


Figura 3 – Diagrama de blocos do sistema.

- **Software Energia:**

Para os primeiros testes do protótipo, utilizou-se a ferramenta “Energia” com a finalidade de desenvolver o software embarcado.

Para o controle dos servomotores utilizou-se a biblioteca “Servo.h” e para a leitura do sensor a leitura serial do MSP.

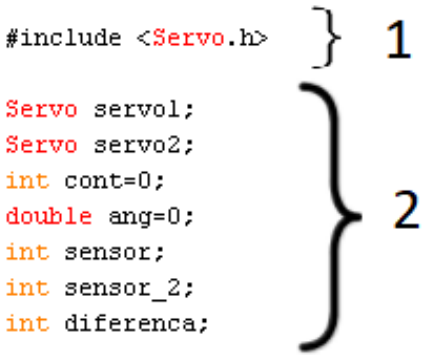


Figura 4 – Descrição do software, primeira parte.

Na parte 1, indicada na figura 4, é incluído a biblioteca utilizada para o servomotor.

Em 2, é definido as entidades dos 2 servomotores, além da declaração das seguintes variáveis: “cont” que será respnsável pela contagem de pulsos (forçamento do músculo), “ang” que armazenará o valor da angulação a ser passado para o servo,“sensor” responsável por armazenar o valor analógico do sensor no primeiro instante de tempo, “sensor_2”, que armazenará o valor analógico do sensor em um segundo instante de tempo, e “diferenca” que armazenará a diferença dos valores analógicos dos sensores em instantes de tempo distintos.

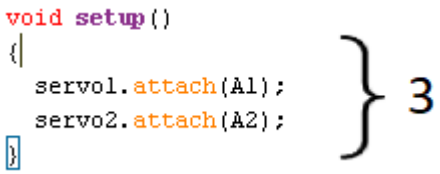


Figura 5 – Descrição do software, segunda parte.

Na segunda parte do código são feitas as configurções iniciais do sistema. Em 3, indicado na figura 5, são instanciados os pinos do MSP430 aos servomotores.

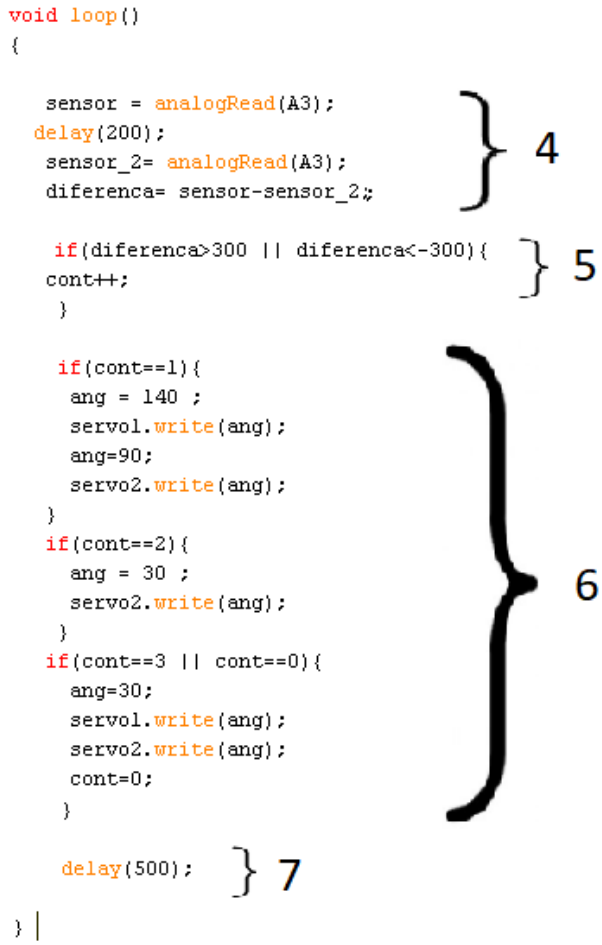


Figura 6 – Descrição do software, segunda parte.

A partir da parte mostrada na figura 6, o código entra em um loop, em que é constantemente repetido.

Em 4 as variáveis “sensor” e “sensor_2” recebem o valor de leitura analógica do piezoelétrico no primeiro instante de tempo e após 200ms, respectiamente. Após a leitura dos valores, a variável “diferenca” recebe a diferença entre os valores de “sensor” e “sensor_2”. Essa operação de diferença é realizado com o intuito de detectar apenas contrações rápidas do braço, apenas quando ocorre a intenção do usuário, e para não detectar o movimento natural do braço, como por exmplo de levanda-lo e abaixa-lo.

Em 5 é definido uma estrutura condicional, em que se o valor da variável “diferenca” for maior que 300 ou menor que -300 , que representa o momento em que o braço é contraído, é somado 1 a “cont”.

Em 6, são definidos 3 estruturas condicionais, responsáveis por acionar o servo, e consequentemente o movimento da prótese. A primeira condição é acionada caso o valoro da variável “cont” seja igual a 1, passando a angulação de 140º para o primeiro servo e de 90º para o segundo servo, realizando o movimento de fechar a mão. A segundo condição é acionada caso “cont” seja igual a 2, e atribui 30º ao segundo servo enquanto o pimeiro permanece com a mesma angulação, realizando com isso, o sinal de rang loose. Por último, a terceira condição é acionada caso “cont” seja igual a 3, atribuindo 30º aos dois servomotores, realizando o movimento de abrir a mão.

Em 7 é realizado um atraso de 500ms, com o intuito de providenciar o um tempo para o acionamento e posicionamento dos servomotores antes do próximo movimento. Além disso, possui a função de não detectar falsos acionamentos, ou seja, minimizar o erro, em casos de o usuário contrair, indesejadamente ou por acidente, o músculo mais de 1 vez entre cada movimento da prótese, e para que o valor do sensor se estabilize entre cada leitura. Sem o “delay” o software poderia interpretar o sinal de contração muscular mais de uma vez antes da estabilização do valor do sensor, ou seja, a variável “cont” poderia ser incrementada mais de uma vez em apenas uma contração do braço.

- **Software Code Composer:**

Após validar os primeiros testes no software “Energia”, migrou-se para o software “Code Composer”, devido à possibilidade de configurar o código instrução por instrução.

O código inicia-se com leitura analógica do sensor piezoelétrico, realizada a cada 1 milissegundos. O valor de tensão, gerado pelo sensor, é convertido em um sina digital de valor 0, quando 0V, e 1023, qunado 3,6V. Para a converçã A/D utiliza-se da função ADC10 do msp.

Após a conversão A/D, é feito a configuração das saídas PWM utilizadas para o controle dos servomotores. O servo motor utilizado é controlado por um sinal com frequência de 50Hz e pulsos com largura de 5 a 25ms, com a variação da largura de pulso entro desses limites é possível alterar a posição do servo de 0º a 180º.

$$pwm = 100 * \left(\frac{20 * Valor\ lido(sensor)}{1023} + 5 \right)$$

Equação 1 – Cálculo do tempo do pulso

O tempo é multiplicado por 100 para que o valor seja dado em milisegundos, já que o clock do msp está configurado em 1MHZ.

Para que seja realizado a contagem de tempo que será enviado a cada pulso, utilizou-se a comparação do timer em modo “OUTMOD_7”, em que os dutys cycles desse pulso duram de acordo com a equação 1.Para isso, configurou-se TIMERs em modo “up” (MC_1)

Além disso, configurou-se um botão responsável por regular a sensibilidade do sensor. Cada vez em que o botão é acionado o valor da variável “sensibilidade” é diminuido em 150, ou seja, é necessário uma menor pressão sobre o sensor para o acionamento do servo à 180º. Quanto menor o valor de “sensibilidade” menor o esforço necessário sobre o sensor, funcionando de acordo com a equação 2.

$$pwm = 100\left(\frac{ValorRecebido * 20}{sensibilidade} + 5\right)$$

Equação 2 – Cálculo do pwm de acordo com a sensibilidade e com o valor do sensor.

A regulagem de sensibilidade pode ser feita 4 vezes, representado pela variável “nível”, em que o valor de “sensibilidade” varia de 1023 à 423. Ademais, abilitou-se leds para indicar a seleção de cada nível, para o nível 1 é aceso apenas um led, para o nivel 2 é aceso 2 leds, e assim sucessivamente. Além disso, é adicionado um loop usando a função “for” para funcionar com dbounce do botão.

```
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{

   sensibilidade = sensibilidade - 150;
   nivel = nivel + 1;

   long int i;
   for(i=0;i<=30000;i++);

   if(nivel == 1){
          P1OUT |= BIT4;
          P1OUT  &= ~ BIT6;
          P1OUT  &= ~ BIT7;

      }
   if(nivel == 2 ){
          P1OUT |= BIT4;
          P1OUT |= BIT6;
          P1OUT  &= ~ BIT7;

      }
   if(nivel == 3){
          P1OUT |= BIT4;
          P1OUT |= BIT6;
          P1OUT |= BIT7;

      }

    if(nivel == 4){
        nivel = 0;
        sensibilidade = 1023;
        P1OUT  ^= BIT4 + BIT6 + BIT7 + BIT8;

       }

   P1IFG &= ~BIT3;
}
```

Figura 7 - Interrupção associada ao botão.

O valor analógico lido pelo sensor é muito instável, ou seja, possui uma grande variação a cada instante de tempo, e como esse valor é utilizado para o cálculo do pwm, os sevos ficavam muito instáveis e tremiam muito. Para resoslver esse problema, criou-se um vetor de 4 posições, em que o valor lido do sensor é armazenado na posição 0 em cada instante de tempo. Depois do armazenamento é tirado a média dos valores das 4 posiçoes do vetor e é passado para o cálculo do pwm, de acordo com a figura 8.

```
while(1)
      {


          ADC10CTL0 |= ENC + ADC10SC;

          parcial[3]=parcial[2];
            parcial[2]=parcial[1];
            parcial[1]=parcial[0];
            parcial[0]=ADC10MEM;

          ValorRecebido= (parcial[3]+parcial[2]+parcial[1]+parcial[0])/4;

          TACCR1=motor(ValorRecebido, sensibilidade);


                              }
```

Figura 8 – Código de estabilização dos servos.

VIII. DESCRIÇÃO DA ESTRUTURA

Para a confecção dos dedos, utilizou-se de mangueiras com 10mm de diâmetro e 1mm de espessura, e outra com 6mm de diâmetro e 1mm de espesura. Para a estrutura utilizou-se uma tampa de plástico para a produção do suporte, onde os servos e as mangueiras são fixadas por meio de abraçadeiras de nylon. Já para a ligação dos dedos com os servos, são utilizados linhas de nylon.

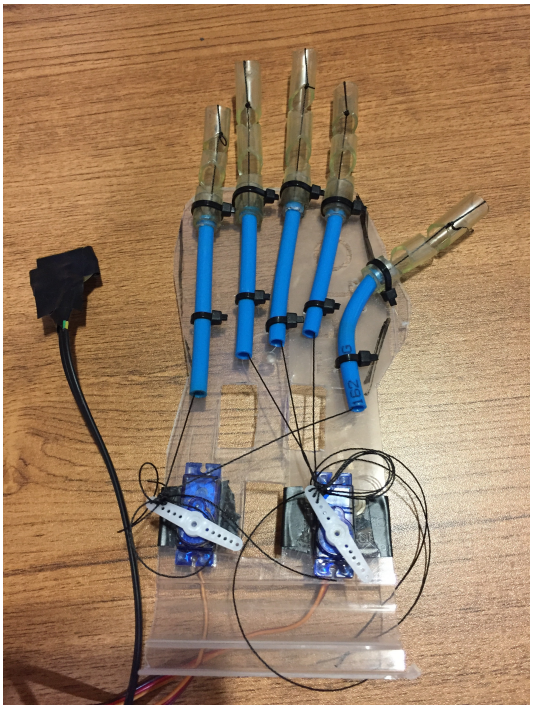


Figura9 - Estrutura do projeto.

IX. CONCLUSÕES E POSSÍVEIS MELHORIAS

- Contrução de uma estrutura de mão mais refinada e melhor;
- Confecção da placa de circuito impresso;
- Montagem do projeto completo;

X. REFERÊNCIAS

[1] http://labdegaragem.com/profiles/blogs/tutorial-como-utilizar-o-piezo-element-com-arduino

[2] http://www.arduinoecia.com.br/2013/06/controlando-um-servo.html

Apêndices

1. Código implementado no msp

```
#include <msp430.h>

unsigned int nivel=0;
unsigned int sensibilidade=1023;
int parcial[4]={0, 0, 0, 0};

int motor(unsigned int x, unsigned int y)
{
  int pwm;
  pwm = 100*((x*20/y) + 5);
  return pwm-1;
}


int main(void)
{

  unsigned int ValorRecebido;
  WDTCTL = WDTPW + WDTHOLD;
  ADC10CTL1 = INCH_5 + ADC10DIV_3 ;
  ADC10CTL0 = SREF_0 + ADC10SHT_3 + ADC10ON;
  ADC10AE0 |= BIT5;
  P1SEL |= BIT5;
  BCSCTL1 = CALBC1_1MHZ;
  DCOCTL = CALDCO_1MHZ;
  P1OUT = 0;
  P1DIR |= BIT2 + BIT0+ BIT4 + BIT8 + BIT6 +
BIT7 ;
  P1REN |= BIT3;
  P1OUT |= BIT3;
  P1IE |= BIT3;
  P1IES |= BIT3;
  P1IFG &= ~BIT3;
  P1DIR |= BIT2;
  P1SEL |= BIT2;
  P1SEL2 &= ~BIT2;
  TACCR0 = 20000-1;
  TACCTL1 = OUTMOD_7;
  TACTL = TASSEL_2 + ID_0 + MC_1;
  _BIS_SR(GIE);

  while(1)
      {


   ADC10CTL0 |= ENC + ADC10SC;

parcial[3]=parcial[2];

parcial[2]=parcial[1];

parcial[1]=parcial[0];

parcial[0]=ADC10MEM;

ValorRecebido=
(parcial[3]+parcial[2]+parcial[1]+parcial[0])/
4;


TACCR1=motor(ValorRecebido, sensibilidade);


}


}


#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{


   sensibilidade = sensibilidade - 150;
   nivel = nivel + 1;

   long int i;
   for(i=0;i<=30000;i++);

   if(nivel == 1){
          P1OUT |= BIT4;
          P1OUT  &= ~ BIT6;
          P1OUT  &= ~ BIT7;

      }
   if(nivel == 2 ){
          P1OUT |= BIT4;
          P1OUT |= BIT6;
          P1OUT  &= ~ BIT7;

      }
   if(nivel == 3){
          P1OUT |= BIT4;
          P1OUT |= BIT6;
          P1OUT |= BIT7;

      }
   if(nivel == 4){
```

```
        nivel = 0;
        sensibilidade = 1023;
        P1OUT  ^= BIT4 + BIT6 + BIT7 ;

       }

   P1IFG &= ~BIT3;
}
```