

# Mão Protética

Amauri da Costa Junior  
FGA  
UNB-Universidade de Brasília  
Gama, Brasil  
Amauri.cj@hotmail.com

Thássio Gabriel Farias dos Santos  
FGA  
UNB-Universidade de Brasília  
Gama, Brasil  
Thassio.96@gmail.com

**Resumo**—Este documento mostra de forma detalhada a elaboração do projeto de uma mão protética controlada pelo microcontrolador MSP430 ao descrever toda a sua composição de Hardware e Software.

**Palavras-Chave**—Mão, Prótese, Software, Hardware.

## I. INTRODUÇÃO

A perda de um dos membros do corpo gera algum tipo de dificuldade para a pessoa que o perdeu. Quando o membro perdido é a mão, funções básicas acabam sendo perdidas como, por exemplo, escrever e carregar objetos. Poder recuperar nem que sejam parcialmente essas funções, é sem duvidas algo desejado por aqueles que não conseguem mais executar essas funções e foi pensando nisso que se decidiu elaborar um projeto com foco nesse tema. Para que fosse possível recuperar essas funções, elaborou-se a ideia de criar uma mão protética.

Para controlar a mão tem-se como ideia usar o microcontrolador MSP430 que receberá em sua entrada um sinal analógico da deformação sofrida pelo Piezoelétrico e baseado nesse sinal será controlado de forma devida um servomotor.

## II. DESENVOLVIMENTO

### A. Descrição de Hardware

Para a realização deste projeto foi utilizada a seguinte lista de materiais

Table I. BILL OF MATERIALS

BILL OF MATERIALS		
ITEM	UNIDADE	QUANTIDADE
MSP430G2553 launchpad	CADA	1
PIEZOELÉTRICO	CADA	1
SERVOMOTOR	CADA	1
LINHA DE NYLON	METRO	1
MÃO DE PLASTICO	CADA	1

FITA ISOLANTE	METRO	3
Elástico	CADA	2
JUMPER	CADA	3
PULSEIRA ELÁSTICA	CADA	1

O hardware consiste em uma mão de plástico que possui fios de nylon que funcionam como tendões que são puxados por um servo motor controlado pela MSP430. O servo motor necessita de alimentação externa, pois a MSP430 não possui a corrente necessária para o motor funcionar como desejado. Também é utilizada uma pulseira onde fica o sensor de pressão que receberá dados do pulso do usuário, esses dados são enviados para o microcontrolador que processa esses sinais.

As conexões que compõem a parte eletrônica do projeto foram feitas de acordo com o anexo 1.

Nela, o servo recebe alimentação externa de uma bateria e sua entrada PMW se conecta ao pino 1.2 do servomotor. A leitura do sinal enviado pelo piezoelétrico é feita pela porta 1.5. O botão utilizado é o que já vem instalado na launchpad e é equivalente ao pino 1.3.

### B. Descrição de Software..

O software foi desenvolvido em linguagem C e assembly utilizando-se da biblioteca msp430G2553.h em C e a msp430.h em assembly, para serem utilizadas palavras chaves que se referem aos registradores, e clocks da msp430g2553.

O código em C possui um laço de repetição que lê o valor de tensão devolvido pelo sensor piezoelétrico, essa leitura é realizada a cada 1 milissegundos. Esse valor de tensão é transformado em valores que vão de 0 a 1023 pelo programa, com 0 sendo 0 volts, e 1023 sendo 3.6 volts. O programa então converte esse valor de tensão em um valor de tempo, pois o motor é controlado pela quantidade de tempo que este recebe tensão.

O valor 0 é transformado em 5 milissegundos e faz com que o motor gire para a posição de 0° e o valor de 1023 é transformado para 25 milissegundos e gira o motor para aproximadamente 180°

O código em C também possui um interrupt que ocorre quando o botão da msp é apertado. Ele serve para calibrar o equipamento, mudando o seu funcionamento para cada tipo de usuário. Se a deformação do piezo não for suficiente para gerar 3.6 volts, ou seja, gerar um valor de 1023, a função não conseguirá devolver 25ms, pois o calculo funciona de acordo com a seguinte equação:

Equação 1. Equação do tempo

$$Tempo = 100 * (\frac{20 \times Valor\ Recebido}{1023} + 5)$$

Pensando nisso, inseriu-se uma segunda variável que diminui o divisor para que tempo do pulso consiga ser 25ms mesmo que a deformação não seja suficiente.

Equação 2. Equação do tempo com calibrador

$$Tempo = 100 * (\frac{20 \times Valor\ Recebido}{Valor\ Calibrado} + 5)$$

O tempo é multiplicado por 100, pois como a o clock da msp está configurado como 1MHz, e para que o atraso ocorra na faixa dos milissegundos, é necessário ser multiplicado por 100.

O Valor Recebido é o valor dado pela leitura analógica.

O Valor calibrado é um valor que começa em 1023 e toda vez que o botão é apertado ele é reduzido em 100 até que chegue em 423, ao chegar nesse valor ele é resetado para 1023.

Para que seja feita a contagem do tempo em que será enviado cada pulso, utilizou-se a comparação de timer no modo OUTMOD\_7, quando o timer conta até 20 milissegundos, ele reseta a contagem. Os dutys cycles destes pulsos dura de acordo com o Tempo visto na equação 2.

Todo o código em C pode ser visto no anexo 3.

O código em assembly foi feito como uma tradução do código em C, funcionando exatamente do mesmo jeito que o código em C, entretanto o interrupt no assembly não funcionou corretamente.

Tentou-se juntar os códigos, colocando alguma das funções de assembly junto com o resto do código em C, porém foram encontrados problemas, e o código nunca funcionava corretamente, por isso o código final foi feito todo em C.

### III. RESULTADOS

O conjunto hardware e software foi analisado e testado para se obter os resultados necessários, a pulseira foi colocada no braço da pessoa que iria realizar os testes, e então foi o hardware foi calibrado para o usuário, então foram realizados

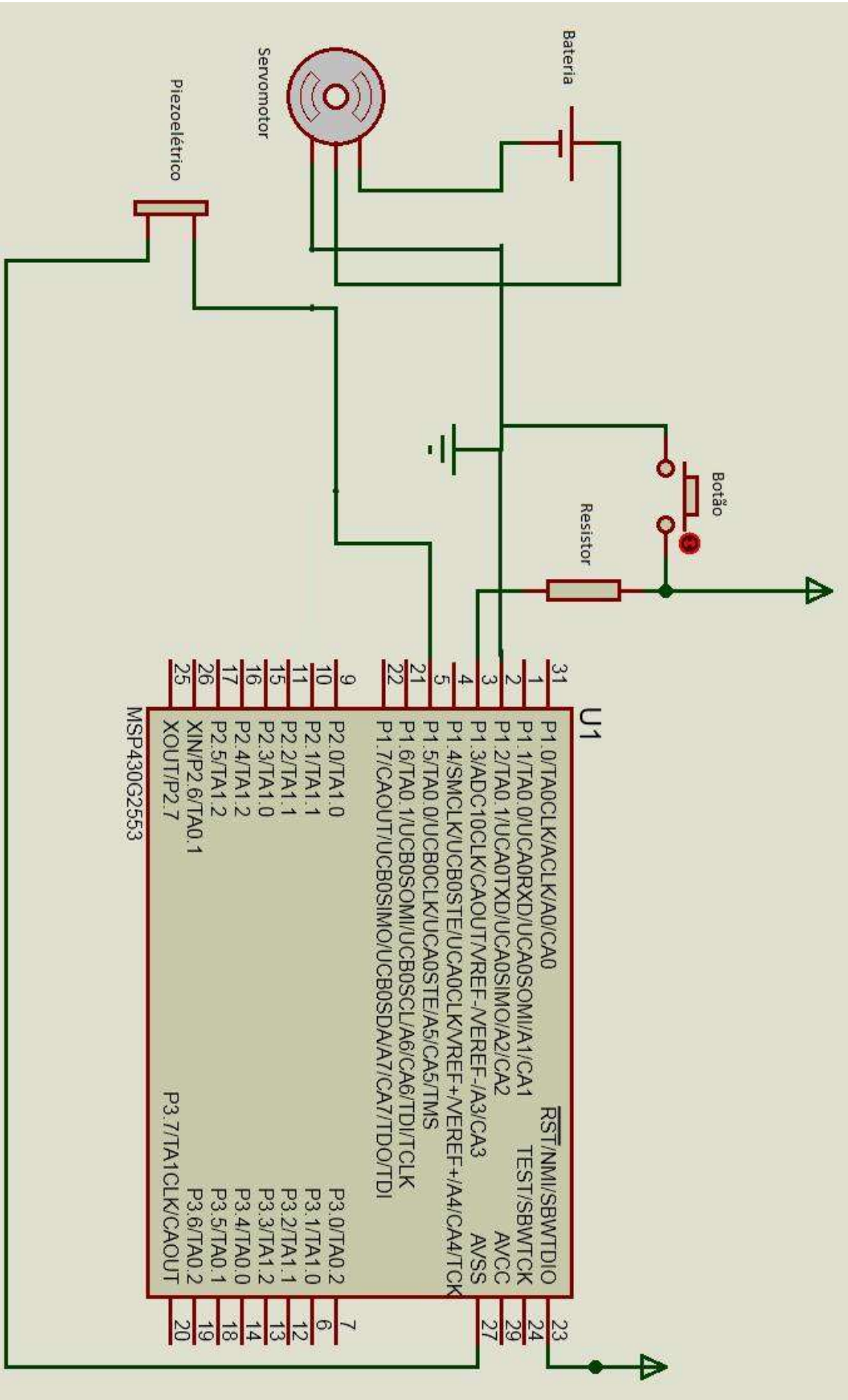
movimentos de abrir e fechar a mão para ver se a mão protética respondia corretamente.

A mão respondeu muito bem ao movimento da mão real, os dados obtidos pela pulseira são corretamente traduzidos para tempo no software e depois em movimento do motor.

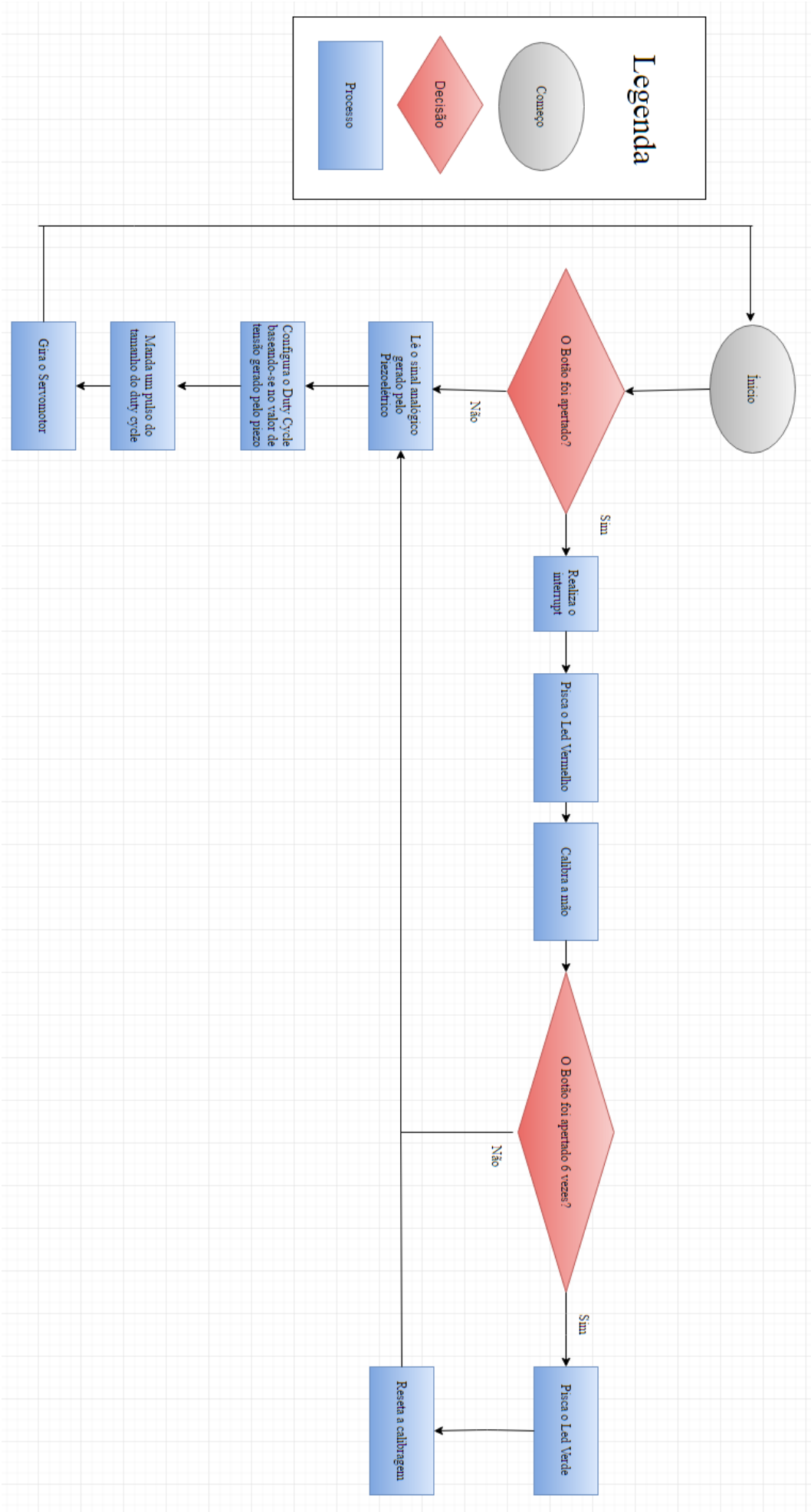
### IV. CONCLUSÃO

Após a finalização do código, e a montagem da mão protética, pode-se concluir que tudo funcionou como o esperado de acordo com o que foi proposto, que era uma mão protética que repetiria os movimentos de abrir e fechar de uma mão humana. Apesar de alguns problemas, como a falta do assembly no código, e a falta de força na mão quando ela tem que voltar pra posição inicial, o conjunto hardware e software funcionou muito bem, lendo corretamente os valores vindos do pulso do usuário, e traduzindo esses valores para movimento do motor e consequentemente da mão protética.

Anexo 1 – Esquemático de montagem do projeto



Anexo 2 – Fluxograma do funcionamento do código



### Anexo 3 – Código em C

```
#include <msp430g2553.h>

unsigned int conta = 0;
unsigned int calibrador = 1023;

int regrade3cal(unsigned int x, unsigned int c)
{
    int pwm;
    pwm = 100*((x*20/c) + 5);
    return pwm-1;
}

void main(void)
{
    {
        unsigned int ValorRecebido;
        int ValorAtraso;
        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 + BIT6;
        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)
        {
            __delay_cycles(1000);
            ADC10CTL0 |= ENC + ADC10SC;
            ValorRecebido = ADC10MEM;
            TACCR1 = regrade3cal(ValorRecebido, calibrador);
        }

    }

    #pragma vector=PORT1_VECTOR
    __interrupt void Port_1(void)
    {
        long int o;
        P1OUT |= BIT0;
        for(o=0;o<40000;o++);
        calibrador = calibrador - 100;
        conta = conta + 1;
        P1OUT ^= BIT0;
        if (conta == 6){
            conta = 0;
        }
    }
}
```

```
calibrador = 1023;
P1OUT |= BIT6;
for(o=0;o<40000;o++);
P1OUT ^=BIT6;
}
P1IFG &= ~BIT3;
}
```

#### Anexo 4 – Subrotina em assembly

##### ; FUNÇÃO REGRA DE 3

Regrade3:

mov.w #17, R14

call #Multiplicacao

mov.w R4, R14 ;R4 é o valor d

calibrador

mov.w R15, R8

call #Divisao

add.w #5, R10

jmp FIMMESMO

;R14 multiplicador

;R13 numero que ta sendo multiplicado

;R15 resultado

Multiplicacao:

mov.w R13, R15

mov.w R14, R12

cmp #0, R14

jeq CondiZero

cmp #1, R14

jeq FIM

Somas:

add.w R13, R15

cmp #2, R12

jeq FIM

dec.w R12

jmp Somas

CondiZero:

mov.w #0, R15 ;Resultado em R15

jmp FIM

FIM:

clr.w R12

Ret

Divisao:

cmp R14, R8

j1 Condi1

cmp #1, R14

jeq Condiacao1

mov.w R14, R11

sub.w R14, R8

add.w #1, R12 ; R8/R14

Comparar:

dec.w R11

cmp R11, R8

jeq Resultado

cmp #0, R11

jeq Divisao

jmp Comparar

Condiacao1:

mov.w R8, R12

jmp Resultado

Condi1:

mov.w #0, R12

jmp Resultado

Resultado:

mov.w R12, R13

mov.w R14, #100

call #Multiplicacao

mov.w R12, R10;

clr.w R14

clr.w R12

clr.w R11

clr.w R8

ret