

Prótese Eletrônica Auxiliar Infantil

Tiago Pereira, *Engenharia eletrônica, UNB*, Gabriel Genari, *Engenharia eletrônica, UNB*

Keywords—prótese, biomédica, infantil.

I. RESUMO

A Proposta deste artigo é de abordar previamente o planejamento e montagem de uma prótese eletrônica projetada para faixa etária de 3 e 7 anos de idade com a função de segurar. Neste primeiro ponto de controle temos a proposta de demonstrar previamente como funcionará o projeto utilizando os métodos propostos em aula.

II. INTRODUÇÃO

A utilização das próteses ortopédicas são datadas dos tempos mais antigos, sendo o registro mais antigo na Índia Antiga entre 3800 a.C e 1400 a.C [1]. O desenvolvimento delas evoluiu de forma exponencial no último século com a evolução da área biomédica utilizando componentes eletrônicos, alcançando funções de alta complexidade como mecanismos telemétricos eletrônicos de alta complexidade baseados nos joelhos humanos [2].

A grande dificuldade é que grande parte das próteses com funções eletrônicas tendem a preços inacessíveis para a maior parte da população brasileira, onde uma prótese de alta capacidade esteja entre os valores de R\$46.000,00 e R\$190.000,00 [3].

Especificamente a população brasileira de deficientes motores com grande dificuldade é constituída por 3.698.929 pessoas, ou cerca de 2% da população brasileira total[4].

Representando um número expressivo de pessoas onde apenas 3%, cerca de 110.000, teriam alguma condição para utilizar alguma prótese de funções eletrônicas [5]. Acrescido a estes fatos e ao analizarmos a população infantil, entre 0 e 9 anos, temos um número muito menor de deficientes motores que a população adulta, devido principalmente a uma parte grande dos deficientes não nascem com ela mas adquirem após algum trauma que compromete

as funções motoras [4]. Ocasionalmente uma eventual diminuição no investimento em próteses complexas nesta faixa etária, mesmo sendo as mais críticas para desenvolvimento das funções básicas.

Tornando assim a proposta de uma prótese de baixo custo com funções eletrônicas básicas para faixa etária de 3 a 7 anos uma proposta viável de implementação.

III. METODOLOGIA

Para o planejamento deste projeto foi utilizado a metodologia 5W2H utilizando as abordagens de TOP-DOWN e BOTTOM-UP organizadas em um quadro de planejamento, descrito nas seções posteriores.

Primeiramente foi definidos os requisitos básicos e funções básicas que a prótese irá incorporar. Assim como os objetivos, ameaças, custos, cronograma de entrega conforme descrito posteriormente neste artigo.

Em seguida será realizado um esboço técnico no programa CATIA do formato e das peças constituintes da prótese, incorporando as medições reais dos componentes eletrônicos e mecânicos necessários para realização das funções definidas para a prótese.

Com o desenho técnico final realizado e as peças finais definidas ocorre a compra dos materiais e impressão 3D das partes necessárias. Em paralelo a este processo temos a prototipagem de um modelo preliminar de mecanismo imitando a mão humana para a função de segurar, mais detalhadamente descrito em seguida, utilizando o microcontrolador MSP430FR2433 na plataforma Energia de programação.

Posteriormente a finalização das peças utilizando impressão em 3D e a chegada dos materiais ocorre a montagem final da prótese e melhoramento os códigos utilizados no controle no microcontrolador, conhecimento este adquirido durante as aulas de microcontroladores.

Paralelamente a este processo ocorre a correção dos erros e adaptações necessárias para funcionamento do protótipo e implementação das dicas fornecidas pelo professor Diego Caetano Garcia.

IV. OBJETIVOS

O objetivo deste projeto é o planejamento e prototipagem de uma prótese de substituição funcional para deficientes físicos por amputação maior em um nível pouco acima do cotovelo ou pouco abaixo do cotovelo (quase total do antebraço) com faixa etária entre 3 e 7 anos no braço direito. Com o intuito de realizar funções básicas de pegar objetos e apoiá-los, por crianças em estágio de desenvolvimento cognitivo inicial. Além de proporcionar uma adaptação a próteses na infância, aumentando a aceitação e utilização de tais produtos na vida adulta. Suas funcionalidades serão controladas e ajustadas eletricamente por um microcontrolador MSP430FR2433, projetado e vendido pela *Texas Instruments™*.

V. VANTAGENS E DESVANTAGENS

A. Vantagens

a) : Os movimentos proporcionados pela prótese são superiores as próteses de mercado pois não possui apenas uma função estática, tornando a vida da criança mais dinâmica.

b) : Utilizando a prótese desde a infância, a adaptação a novas próteses na adolescência e vida adulta é facilitada.

c) : O uso desta prótese diminui a dificuldade da criança ao entrar no meio escolar, pois anatomicamente não terá a falta de um membro do corpo comparado as sem deficiência.

d) : Diminui a necessidade de ajuda de outras pessoas em certas tarefas que necessitem duas mãos para carregar algum objeto, aumentando a autonomia da criança.

B. Desvantagens

a) : Como a prótese necessita de baterias, ela precisa ser carregada antes do uso e possui uma autonomia limitada se comparada a uma prótese simples.

b) : Mesmo com uma característica básica de ser mais barato existirá um custo envolvido para compra da prótese, diminuindo a acessibilidade em famílias de extrema pobreza.

c) : Dependendo da deficiência da criança o braço pode não ser compatível sua anatomia, ou desconfortável ao ponto de rejeição portador.

VI. REQUISITOS

a) : A função básica proposta é a de agarrar objetos e segurá-los de forma uniforme com a tensão necessária para que ele não caia e também não o danifique com tensão excessiva. Os principais objetos que devem ser suportados pela prótese são brinquedos do usuário além de objetos de uso pessoal como mamadeiras e escovas de dente.

b) : Para a movimentação dos dedos será desenvolvido um sistema de molas que, com o auxílio de motores e fios, dobrará os dedos tornando a manipulação dos objetos possível.

c) : O design da mão e do braço deve ser viável para impressão 3D e possuir uma familiaridade com a forma anatômica que se baseiam [6].

d) : A movimentação dos dedos deve ser o mais natural possível, evitando a rejeição da criança que possui um braço normal e outro com prótese [7][8].

e) : O controle dos dedos da mão desta prótese para cada função deve ser eficiente, evitando uma perda energética e a eventual necessidade de recarregar a bateria ao seu esgotamento.

f) : Utilizar um boneco de tamanho similar ao de uma criança definidas no escopo do projeto para realização de teste guiado pelo projetistas para as funções propostas.

VII. AMEAÇAS

a) : A utilização da prótese proposta não será viável se o custo-benefício não for competitivo, dessa forma deve-se priorizar o mínimo custo com as funcionalidades propostas.

b) : A utilização de material de baixa qualidade na fabricação das peças pode comprometer a segurança do produto, soltando partes que proporcionam perigo de ingestão pelas crianças usuárias da prótese, que pode ser solucionada utilizando um material mais maleável e atoxico que não apresenta riscos de quebra.

c) : A falha de distribuição de energia para o circuito pelo esgotamento da bateria precocemente inviabiliza a função básica da prótese, portanto para ampliação do período de operação é necessário a otimização do software empregado pela utilização do modo de baixo consumo no microcontrolador MSP430FR2433.

VIII. CRONOGRAMA

O cronograma, na tabela I, para o projeto se baseia nas datas propostas pelo professor da disciplina para os pontos de controle, acrescido da data dos testes e possíveis semanas para discussão do projeto.

Tabela I. CRONOGRAMA PRELIMINAR PARA O PROJETO

Dias da semana	Atividade	Dias da semana	Atividade
	Março	13/05 à 19/05	Prova 2 (16/05)
18/03 à 24/03	Pesquisa da Prótese	20/05 à 26/05	Aprimoramento do Protótipo
25/03 à 31/03		27/05 à 02/06	Ponto de controle 3
	Abril	Junho	
01/04 à 07/04	Ponto de controle 1º (04/04)	03/06 à 09/06	
08/04 à 14/04	Desenho técnico no CATIA e programação inicial dos subsistemas de controle em C	10/06 à 16/06	Ponto de controle 4 (13/06)
15/04 à 21/04	Teste 1º (18/04)	17/06 à 23/06	
22/04 à 28/04	Impressão 3D das peças para prótese, montagem e teste de funcionamento	24/06 à 30/06	Entrega Final
	Maio	Julho	
29/04 à 05/05	Desenvolvimento do relatório	01/07 à 07/07	
	Ponto de controle 2º (02/05)		
06/05 à 12/05	Refinamento dos códigos e teste de durabilidade do protótipo	08/07 à 14/07	

IX. PROTÓTIPO FUNCIONAL - PONTO DE CONTROLE 2

Conforme definido pelos requisitos básicos e objetivo do projeto, nas seções VI e IV respectivamente, será necessário que os componentes exigidos para o protótipo simulem a função de agarrar para a prótese. Com esse intuito foi definido os componentes básicos para o protótipo inicial, tabela II, funcionando de forma conjunta ao microcontrolador MSP-EXP430FR2433LP, figura 1, este que ambos projetistas possuem e atendem aos requisitos mínimos.

Tabela II. MATERIAIS PARA PROTÓTIPO FUNCIONAL

Quantidade	Materiais
2	Microcontroladores MSP-EXP430FR2433LP
1	Sensor de força resistivo Sparfun
2	Sensores de distância Ultrassônico - HC-SR04
1	Display de OLED 0.96"
1	i2C - SSD1306
2	Servomotor TowerPro 9g SG90
1	Garra de Acrílico
Variado	Jumpers
Variado	Massa Cerâmica
1	PowerBank 5V & 1/2.4A
1	Fio com conector USB tipo A
Variado	Fio de Cobre 34

Para o protótipo funcional foi utilizado a plataforma de SDK Energia, criada especificamente para utilização das bibliotecas do Arduino no MSP430. O protótipo é dividido em duas partes, controladas cada uma por um microcontrolador MSP430FR2433, figura 1:

- 1) Demonstrar a movimentação dos dedos da mão utilizando servomotores, com uma maquete simples do formato da mão;
- 2) Demonstrar o funcionamento dos sensores de ultrassom, mostrando a distância entre 2-9cm em funcionamento paralelo na tela de OLED, e o sensor de força resistivo, utilizando o monitor serial do SDK para mostrar a força aplicada ao sensor.

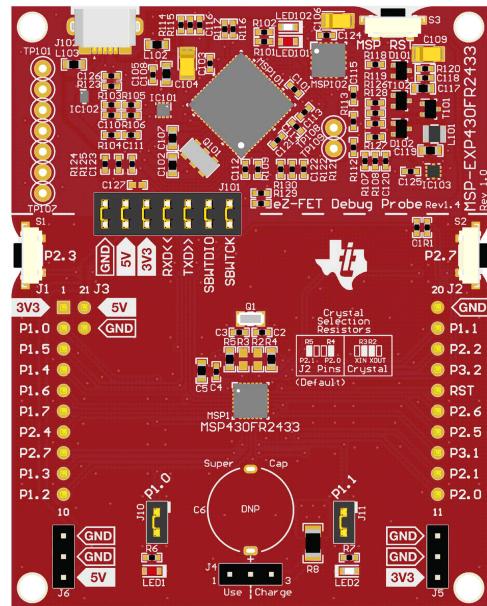


Figura 1. Microcontrolador LaunchPad MSP430FR2433LP

A seguir temos o funcionamento de cada um dos códigos respectivamente para cada uma das partes.

A. Parte 1 - Teste da movimentação dos dedos usando servomotores

O objetivo deste teste é demonstrar o funcionamento dos servomotores para controle dos dedos de uma prova de conceito do mecanismo da mão de uma prótese. O protótipo conceitual do mecanismo foi modelado a mão utilizando massa cerâmica, com as juntas sendo feitas de pedaços de tubos para instalação elétrica e movimentados por fios de cobre 34 conectados a servomotores do tipo descrito na tabela II.

O código, apêndice "Código do protótipo funcional- seção servomotores, resumidamente controla os servomotores em uma rotina utilizando botões para teste de abertura e fechamento da mão. Ao clicar 1 vez no botão touchscreen o servomotor 1 gira para um grau desejado movimentando 1 dedo. Clicando 2 vezes no botão touchscreen todos os servomotores giram para um grau desejado movimentando todos os 5 dedos e, como terceira opção, ao clicar 3 vezes no botão o servomotor que movimenta o dedo polegar se desloca para um grau desejado, movimentando este dedo.

B. Teste dos Sensores

1) Parte 2 - Sensores de distância ultrassônico:

O objetivo deste teste é demonstrar o funcionamento dos sensores de distância ultrassônico para mostrar suas limitações e aplicações para o projeto. São utilizados dois sensores, um para medição da distância de um objeto a ser segurado a palma da mão e outro para medição da mão à superfície em que o objeto está apoiado. O código, apêndice "Código do protótipo funcional- seção sensores, resumidamente se baseia no cálculo da distância para centímetros e impressão destes dados na tela de OLED, quando dois objetos estão a uma distância igual ao dos sensores e esta sendo menor de 3 cm ocorre o acionamento da função de fechar a mão (neste caso sendo representado por uma garra controlada por um servo).

2) Parte 3 - Sensor de força resistiva: O objetivo deste teste é demonstrar o funcionamento do sensor de força resistivo para mostrar suas limitações e aplicações para o projeto. São utilizados um sensor de força resistivo e uma resistência equivalente de $3,3k\ \Omega$. O código, apêndice "Código do protótipo funcional- seção sensor de força, resumidamente realiza o cálculo da força com base na resistência equivalente e a fonte de tensão em que o sensor é colocado em série e printa no monitor serial, que pode ser visualizado no Energia, com a resistência em que o sensor possui e a força equivalente que é realizada.

X. RESULTADOS DO PROTÓTIPO CONCEITUAL

Conforme objetivo do protótipo funcional era, para este projeto, uma demonstração inicial de uma maquete inicial do protótipo sendo controlada por servomotores e utilização inicial dos sensores de

distância e de força, a seguir temos o resultado obtido e demonstrado no dia da apresentação do ponto de controle 2.

A. Parte 1 - Teste dos dedos usando servomotores

A primeira parte teve como resultado a maquete inicial feita de cerâmica movimentada por 3 servomotores TowerPro 9g conectados a um botão sensível ao toque e controlados pelo MSP430FR2433LP com o código anteriormente explicado, apêndice "Código do protótipo funcional", carregado em sua memória flash. Todos estes componentes são alimentados por um PowerBank com tensão de saída à 5V e 1A.

A foto do objeto final demonstrado na aula de apresentação apresenta-se abaixo, figura 2.

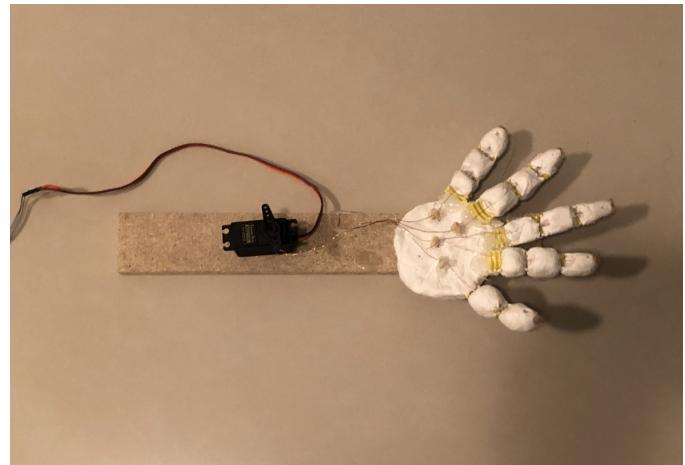


Figura 2. Maquete final com o MSP430 controlando os servomotores

B. Teste dos Sensores

1) Parte 2 - Sensores de distância ultrassônico:

A segunda parte tem como resultado um circuito elétrico com 2 sensores de distância ultrassônicos, uma tela de OLED e um servomotor TowerPro 9g conectados ao MSP430FR2433LP. Estes componentes são alimentados por uma fonte de tomada com tensão de saída à 5V e 2,4A e o circuito elétrico que conecta todos os componentes está demonstrado na figura 8 no apêndice "Esquemáticos".

2) Parte 3 - Sensor de força resistiva: A terceira e última parte tem como resultado um circuito elétrico com o sensor de força resistiva e um resistor de aproximadamente $3,3k\ \Omega$ conectados a um dos conversores A/D do MSP430FR2433LP. Estes são alimentados por uma fonte de tomada com tensão

de saída de aproximadamente 5V e 1A e o circuito elétrico resultante está demonstrado na figura 9 no apêndice "Esquemáticos".

XI. CONCLUSÕES DO PROTÓTIPO FUNCIONAL

Conforme demonstrado na aula demonstrada do ponto de controle 2 para o professor o objetivo do protótipo funcional foi realizado com sucesso. Demonstrando assim que era possível a movimentação dos dedos da maquete utilizando servomotores e o funcionamento correto dos sensores de distância, que mostravam na tela a distância dos objetos em centímetros, e do sensor de força resistiva, que mostrava no monitor serial a resistência interna do sensor e a força correspondente a esta resistência.

XII. DIMENSÕES DA PRÓTESE FINAL - PONTO DE CONTROLE 3

Como descritos anteriormente a principal função para prótese será a de segurar objetos de tamanho suportado para uma criança entre as idades de 3 e 7 anos. Conforme a figura 3, uma criança entre 3 e 7 anos com funcionalidade normal nas juntas dos dedos consegue realizar uma força máxima de aproximadamente 45-60N entre o dedo indicador e o polegar, estes sendo os mais vitais para funcionalidade de segurar.

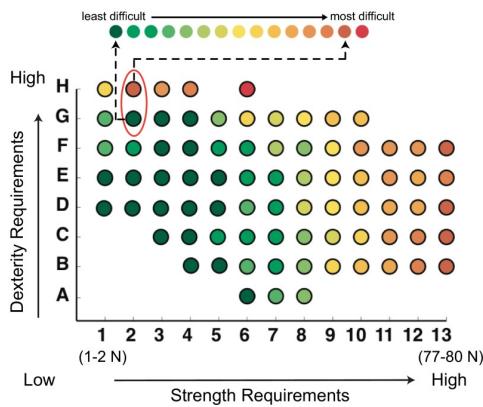


Figura 3. Gráfico de destreza x força realizado entre crianças com 3-13 anos [9]

Com a força máxima necessária definida se faz necessário determinar as dimensões médias do braço de uma criança para escolha dos componentes e modelagem 3D da prótese final. Conforme amostragem realizada no desenvolvimento da prótese Cyborg Beast [10], gráfico da figura 4, podemos definir as

dimensões aproximadas para o braço de uma criança com base em um adulto entre 16-20 anos.

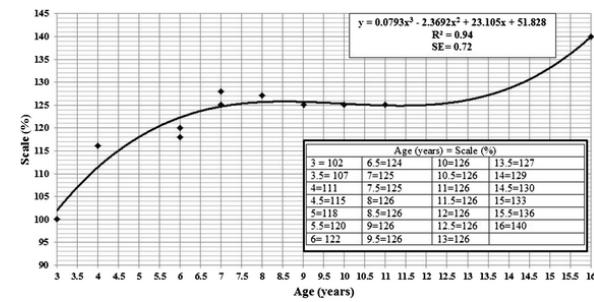


Figura 4. Gráfico mostrando a relação de dimensões da prótese Cyborg Beast em porcentagem conforme a idade [10]

Utilizando como base dos cálculos do gráfico 4 obtemos os valores necessários para a prótese entre as idades almejadas do projeto com base nas dimensões de um dos participantes do projeto, com idade de 20 anos. Estas dimensões, tabela III, representam aproximadamente 140% dos tamanhos médios para uma criança entre 3-4 anos, conforme descrito no gráfico da figura 4.

Tabela III. DIMENSÕES MEDIDAS EM UM PARTICIPANTE DO PROJETO COM IDADE DE 20 ANOS, LEGENDA 5

Dimensão	Medição Participante (cm)	Medida final Criança com 4 anos (cm)	Escala (%)
Comp. A	09,00	06,89	76
Comp. B	19,60	14,98	76
Comp. C	21,30	16,28	76
Comp. D	06,50	04,97	76
Área do corte transversal no comp. C	11,93	09,12	76
Área do corte transversal no comp. E	72,70	55,57	76

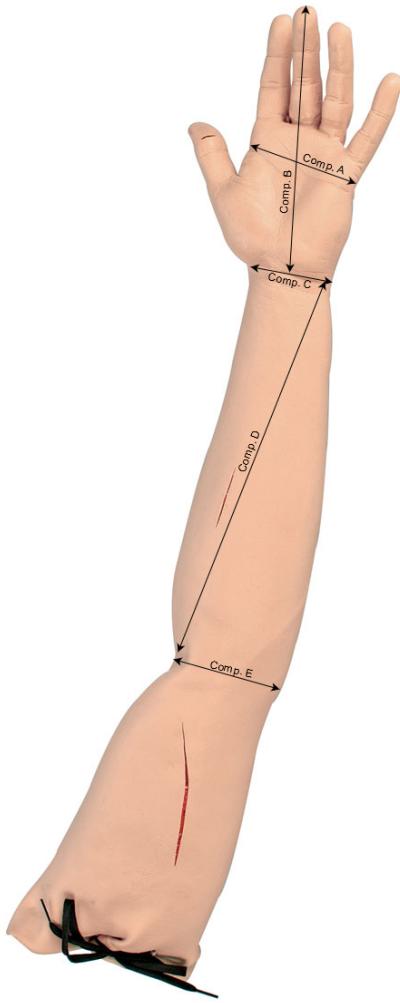


Figura 5. Legenda dos comprimentos medidos na tabela III

Conforme obtido na tabela III as dimensões para a prótese de uma criança com idade de 4-5 anos deve ser de aproximadamente 76% da dimensão de um adulto de 20 anos. Como o intuito deste projeto é demonstrar principalmente a funcionalidade eletrônica, não cabendo uma análise detalhada do desenho e criação 3D do design de um braço completo, foi utilizado o projeto design de braço do projeto MyPo 2.0 [11] com escala reduzida para 75%.

A diferença de escala de 76% para 75% na impressão 3D final é devido as limitações de comprimento que a impressora 3D usada delimitava. Abaixo temos a foto da prótese impressa em 3D em sua primeira versão, figura 6.

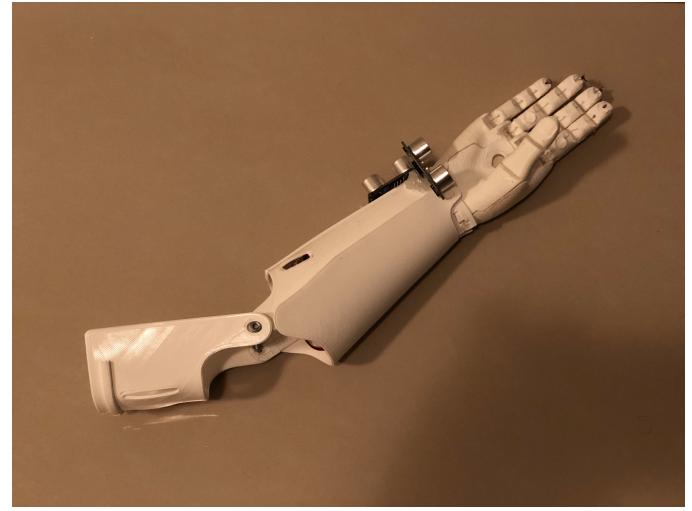


Figura 6. Prótese em 3D para um criança entre 3-5 anos, versão V0 e com base no design MyPo 2.0 [11]

XIII. CONTROLE ELETRÔNICO DA PRÓTESE

A. Componentes e Conexão elétrica

Para a versão final da prótese para este projeto foi utilizado o mesmo tipo de microcontrolador do protótipo funcional, o MSP430FR2433LP. Os demais componentes utilizados podem ser visualizados na tabela IV abaixo.

Tabela IV. MATERIAIS UTILIZADOS NO PROTÓTIPO FINAL

Quantidade	Materiais
1	Microcontrolador MSP-EXP430FR2433LP
1	Sensor de força resistivo Sparfun
2	Sensores de distância Ultrassônico - HC-SR04
2	Resistor de filme de carbono 2,2Ω
2	Resistor de filme de carbono 3,3Ω
2	Servomotor TowerPro 9g SG90
1	Prótese de PLA impressa em 3D
Variado	Jumpers
Variado	Massa de Silicone
1	PowerBank 5V & 1/2.4A
1	Fio com conector USB tipo A

A conexão dos componentes eletrônicos é representada na figura 10 no apêndice "Esquemáticos", esta que demonstra o circuito elétrico final utilizado no projeto.

B. Código de controle

1) PC3: Conforme proposto pelo ponto de controle 3 para o código desta etapa e, consequentemente, uma prévia que como será o código final do

projeto foi utilizado predominantemente bibliotecas (headers) de fábrica do microcontrolador utilizado e o conteúdo absorvido em aula.

Nesta etapa utilizamos o software Code Composer Studio 2017, da Texas Instrumentos (fabricante do microcontrolador), para codificação e debug do código para o protótipo final. Se fazendo necessário, em algumas ocasiões, a utilização do software Energia para o Monitor Serial para teste de alguns sensores.

Primeiramente foi criado um código para controle dos servomotores e separado em duas funções. A primeira, denominada como "servo(volatile int graus)", recebe uma variável int e tem a funcionalidade de movimentar o servomotor para os graus que receber na variável, sendo este um número entre 0 e 180 graus. A segunda, denominada como "Init servos", seta os ports de conexão do pino de controle do servo e o timer A0 no modo de capture/compare. A soma de ambas as funções, no código, cria uma função de controle por PWM no servomotor sendo a função "Init servos()" sendo chamada 1 vez para configuração dos servos e "servo()" sendo chamada em qualquer momento que se quiser movimentar o servomotor para uma posição em graus desejada.

Em seguida ocorreu a criação do código para controle do sensor de distância por ultrassonografia. Para utilização deste sensor primeiro deve-se entender como funciona as entradas de Trigger e Echo, sendo alimentado no pino Vcc e Ground com uma tensão de 5V. o pino de trigger recebe um pulsos de duração a $10\mu s$ que iniciam a funcionalidade do sensor, com o falling edge de 1 pulso do trigger o sensor envia 8 pulsos no pino echo com período de $20\mu s$. Para calcular a distância aplicamos a fórmula para distância em centímetros $D = (T/68) * 1000$, onde T = período total da soma dos 8 pulsos enviados pelo pino echo ao microcontrolador.

Com a lógica de funcionamento captada foi feito o código, função "Init Ultrasensors", utilizando o timer A2 para o 1 sensor e o timer A3 para o sensor 2 e interrupção para a medição em 1 milisegundo. Seu funcionamento é na seguinte forma, os pinos Trigger e Echo são setados como saída e entrada respectivamente; Em seguida é gerado um pulso de $10\mu s$ para o pino de trigger e recebemos o estado para o pino de ECHO, esperando este processo por 30ms; Subsequentemente o código passa por uma interrupção do timer A1 e A2 que ocorre durante a geração dos 8 pulsos para o pino ECHO e calcula

o valor que a distância, este que está na variável sensor; Voltando na função "Init Ultrasensors", que usa o valor da variável sensor, convertemos este valor para centímetros.

Depois criamos uma função atraso utilizando o timer A1, função "atraso(unsigned volatile Tms)". Nesta função o timer é configurado para função de comparação com TACCR0 = 999 = 1000-1, para 1ms com o SMCLK setado a 1Mhz. Em seguida existe um if de decrescimento com a flag TA1IFG resultado da comparação para apenas sair após o número do atraso que deve ser realizado na função, sendo no final o timer A pausado para o próximo delay.

Por fim temos na função principal, "main", um código condicional (if) que segue a lógica do diagrama 7 e tem a principal função do projeto que faz o controle dos servos para função de agarrar da prótese.

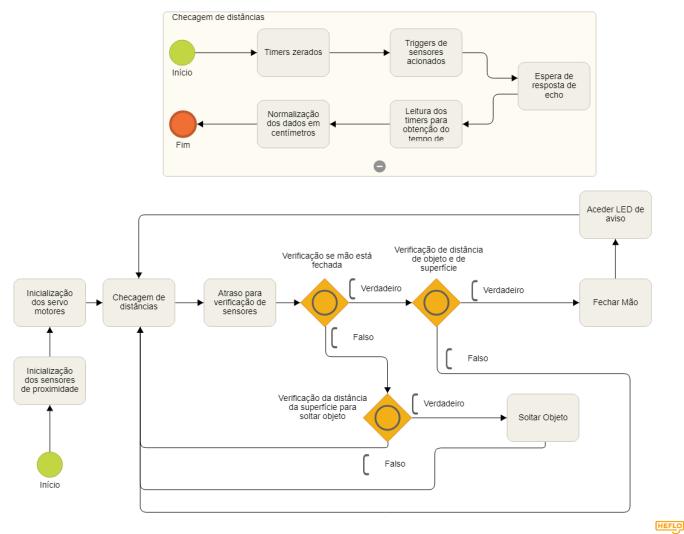


Figura 7. Diagrama de funcionamento da função de agarrar

2) PC4: O código para o ponto de controle 4 é em consequência de um refinamento do código anterior e está representado no apêndice código final PC4. Os refinamentos feitos foram com intuito de adicionar um controle maior aos servomotores, separando os 3 utilizados em 2 canais de comparação do timer A0; Adicionar o sensor de força, conforme definido no começo do projeto, onde pode-se ver a força aplicada à ele em um monitor serial (como no do Energia) conforme se pressiona o sensor e, por fim, o código para envio desses dados via serial utilizando os pinos UART Tx e Rx.

Ao analisar superficialmente o código como um

todo pode-se observar a reestruturação do código para deixar a parte mais importante, a função principal "main", após a definição das variáveis. Com esse intuito no começo foi inserido todas as subrotinas que foram criadas para este código para o compilador dar como existente aquela subrotina, mesmo ela não ter sido ainda definida, para ser usada na função "main". Em ordem, as alterações na função "main" foram mínimas, sendo estas à adição de algumas linhas de código onde iniciamos o sensor de força (Init ForceSensor());, o conversor ADC (ADCCTL0) e a configuração do modo serial UART onde, na subrotina ForceSensorMeasure, ela é utilizadas. Por fim melhoramos o controle dos motores na função condicional if colocando algumas situações a mais que estavam presentes no diagrama 7 mas não foram contempladas no PC3.

Em seguida ocorreu uma alteração na subrotina movimento(volatile int graus) na qual adicionamos a subrotina já descrita antes Init servos();, pois os pinos de seleção P1SEL0 e P1SEL1 são utilizados tanto no UART como nos motores então devem ser zerados. Em seguida temos o clear dos bits de controle para o timer A0, TA0CTL, já que, para economizar energia, ao final desta subrotina desligamos o clock aplicando MC. E, por fim, realocamos a configuração do timer A0, TA0CTL, para está posição pela mesma razão descrita anteriormente. A configuração dos bits de controle para TA0CTL está dentro da subrotina StopServos();.

Logo após temos a subrotina initUart() onde configuramos os bits de controle do UART, UAC0CTLW0, UCA0BR0, UCA0BR1, UCA0MCTLW, UCA0IE, para habilitar interrupção do UART e com um Baud Rate de 9600 para 1Mhz, este valores que foram obtidos no datasheet do microcontrolador MSP420FR2433. Posteriormente temos a subrotina Init Servos() que já foi descrita anteriormente e sofreu alteração para um segundo pino de controle no canal de comparação TA0.2 no pino 2. A subrotina Init Ultrasensors não sofreu nenhuma alteração.

Após estas subrotinas temos a Init ForceSensor que possui a função de setar os bits de controle para o conversor analógico digital de 10 bits para o microcontrolador. Estes são setados para ter como entrada no P1.3 (A3) para o sinal analógico do sensor de força, que em súmula é um divisor de tensão, em uma frequência do subsystem master clock que está a 1Mhz. Seguidamente temos a

subrotina ForceSensorMeasure() que, com base no datasheet da fabricante do sensor e os dados obtidos após conversão ADC, converte para o equivalente da resistência interna existente no sensor de força resistiva e a força resultada no formato de float, string e int. Todas estas que são enviadas utilizando as subrotinas que convertem float, string e int para envio no serial UART.

Por fim temos as interrupções onde as 4 primeiras já foram explicadas e fazem parte do cálculo da distância medida no sensor de distância ultrassônico. Em seguida a interrupção do timer A3 temos a interrupção no conversor ADC, está que foi retirada dos códigos exemplos que a Texas Instruments disponibiliza, e tem a função de sair da interrupção quando a conversão acaba e armazena o resultado, ADCMEM0, na variável ADC Result. Semelhante e retirada da mesma fonte temos a última interrupção que está relacionada ao UART e possui função de esperar a flag UCRXIFG e UCTXIFG para enviarem dados sem problemas de erros.

O código completo se encontra no apêndice "Código final para o PC4", onde se encontram alguns comentários básicos de algumas funcionalidades descritas anteriormente.

XIV. CONCLUSÕES

A. PC3

Conforme demonstrado em sala foi apresentado a prótese impressa em 3D à base de PLA montada e com os sensores de ultrassonografia acoplados em direção ao punho e a mesa, ocorrendo a falta do sensor de força resistiva devido a dificuldades na utilização do conversor A/D no MSP430FR2433, este que é diferente do utilizado em sala. As juntas feitas à base de silicone que junta as partes dos dedos atingiu a proposta porém serão refeitas para uma melhor movimentação dos dedos puxadas pelos fios de linha. Os fios, estes que para esse ponto de controle utilizamos fio de costura normal, atingiram a expectativa que aguentar a força dos servos porém serão trocadas por fios de costura para couro profissional por conter propriedades de acabamento final e resistência à tração muito mais elevadas que o utilizado para essa primeira versão.

Em relação ao controle concluímos que a posição dos sensores a distância funcionou como esperado, não exigindo uma alteração para a prótese final, porém deve ser reescrito a função de controle para

evitar erros de abertura da mão em tempos não desejados, conforme descrito no diagrama 7. E, ao resolver os problemas com o conversor A/D, incluir neste diagrama condições da movimentação do servo com base na força detectada pelo sensor de força resistiva.

Para o design final constatamos a exigência de melhorar o aspecto final da conexão dos sensores e servomotores ao microcontrolador, está que será solucionada após o design e construção de uma PCB customizada. E, por fim, adicionar um suporte para uma fonte de energia 5V 2.4A portátil que seja acoplada ao braço, pois este deve ser portátil pelo fato de ser uma prótese de utilização no dia a dia da criança 3 a 5 anos. O uso dos sensores à distância, conforme proposto pelo projeto, cumpre a ideia desenhada pois é uma prótese mais simplista com intuito de acostumar a criança amputada no uso de próteses.

B. PC4

Como foi apresentado em sala ocorreu a melhora considerável do projeto. As conexões elétricas do microcontrolador com os componentes e a fonte foram simplificadas pois adicionamos uma placa perfurada customizada que funciona como um tipo de Shield para este microcontrolador. Também foi observado o conserto dos bugs ocasionados no código onde a mão não ficava fechada mesmo estando longe de uma superfície e melhora dos cabos que movimentam os dedos da prótese.

Mesmo assim ainda existe algumas pendências antes da entrega final do projeto. Como a construção da PCB customizada para o projeto, impressão 3D de um case que acomode a PCB acoplada ao microcontrolador e a bateria para que o braço se torne portátil, e, por fim, conectar o 3 motor adequadamente no microcontrolador para que este movimente o dedão e dê uma maior aderência ao pegar os objetos.

REFERÊNCIAS

- [1] PARATLETISMO, Braskem. Evolução das próteses na Linha do Tempo. Disponível em: <https://www.braskem.com.br/paratletismo-infografico>. Acesso em: 02 abr. 2018.
- [2] Morris, Beverly A., et al. e-Knee: evolution of the electronic knee prosthesis: telemetry technology development. *JBJS* 83.2_{suppl} (2001): S62-66
- [3] OTTOBOCK. Empresa de próteses. Próteses de Membro Inferior. Disponível em: <https://www.ottobock.com.br/prosthetics/membros-inferiores/>. Acesso em: 02 abr. 2018.
- [4] Demográfico, IBGE Censo. "Disponível em: <http://www.ibge.gov.br>." Acesso em 3 (2010).
- [5] ABOTEC. Associação Brasileira de Ortopedia Técnica. Avaliação de acessibilidade da população para próteses. Disponível em: <http://www.abotec.org.br/novosite/index.html>. Acesso em: 03 abr. 2018.
- [6] Belter, Joseph T., Jacob L. Segil, and BS SM. "Mechanical design and performance specifications of anthropomorphic prosthetic hands: a review." *Journal of rehabilitation research and development* 50.5 (2013): 599.
- [7] Mavani, Rutvij B., Dharmik H. Rank, and Helina N. Sheth. "Design and Working of Myoelectric Prosthetic Arm." *International Journal of Engineering Development and Research* 2.3 (2014): 3324-3333.
- [8] Chiri, Azzurra, et al. "Mechatronic design and characterization of the index finger module of a hand exoskeleton for post-stroke rehabilitation." *IEEE/ASME Transactions on mechatronics* 17.5 (2012): 884-894.
- [9] DUFF, Susan V. et al. Innovative evaluation of dexterity in pediatrics. *Journal of Hand Therapy*, v. 28, n. 2, p. 144-150, 2015.
- [10] ZUNIGA, Jorge et al. Cyborg beast: a low-cost 3d-printed prosthetic hand for children with upper-limb differences. *BMC research notes*, v. 8, n. 1, p. 10, 2015.
- [11] Myo & PO, MyPo 2.0 . "Disponível em: <https://www.thingiverse.com/thing:2409406>". Acesso em 6 Maio (2018).

APÊNDICE IMAGENS

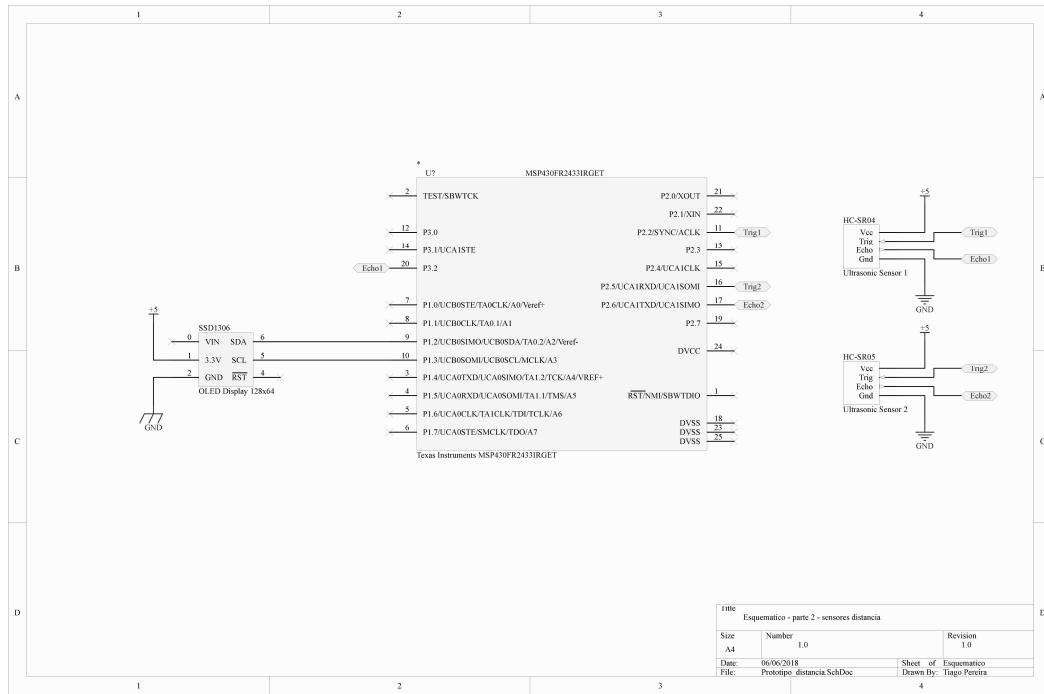


Figura 8. Circuito elétrico para os sensores ultrassônicos

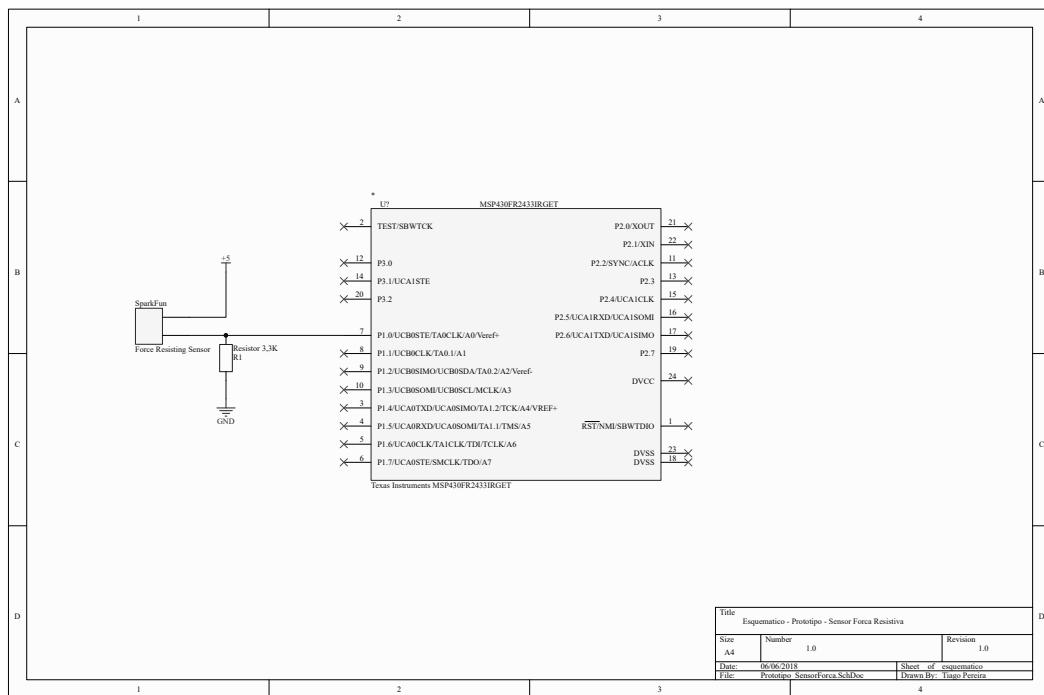


Figura 9. Circuito elétrico para o sensor de força resistiva

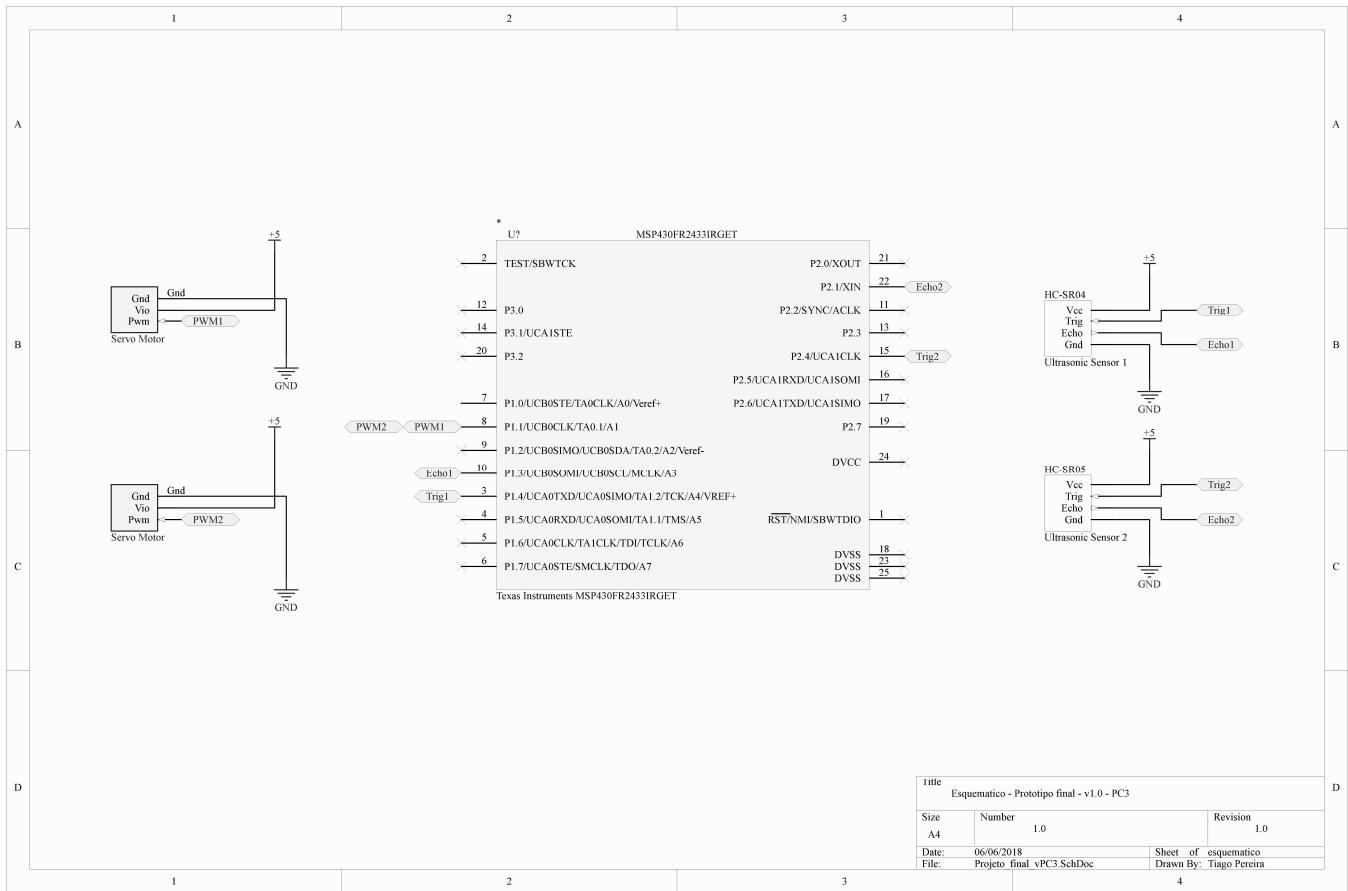


Figura 10. Circuito elétrico utilizado no controle da prótese final do projeto

APÊNDICE

CÓDIGO - PROTÓTIPO FUNCIONAL

A. Parte 1 - Controle Servos - Energia

```
#include <Servo.h>
```

```
#define BTN 10
#define SERVO1 7
#define SERVO2 8
#define SERVO3 9
```

```
Servo myservo1;
Servo myservo2;
Servo myservo3;
```

```
unsigned long tempo=0;
```

```
int contagem=0;
```

```
int aberto=0;
```

```
int btnAnt;
```

```
void setup()
```

```
{
```

```
  pinMode(BTN, INPUT);
  pinMode(2, OUTPUT);
```

```
  myservo1.attach(SERVO1);
  myservo2.attach(SERVO2);

```

```
  myservo3.attach(SERVO3);
  myservo1.write(0);

```

```
  myservo2.write(0);
  myservo3.write(0);
}
```

```
void loop()
```

```
{
```

```
  int btn;

```

```
  btn = digitalRead(BTN);
```

```
  if (btn != btnAnt && btn == 1) {
    contagem++;
  }
```

```
  btnAnt = btn;
```

```
  if(millis()- tempo >= 2000) {
```

```
    switch(contagem) {
```

```
      case 1 :
```

```
        digitalWrite(2,HIGH);
        abreFech(&myservo1);
        contagem = 0;
        break;
```

```
      case 2 :
```

```
        abreFech(&myservo2);
```

```
        contagem = 0;
        break;
      case 3 :
        abreFech(&myservo3);
        contagem = 0;
        break;
      case 4 :
        if (aberto == 0) {
          aberto = 1;
          myservo1.write(170);
          myservo2.write(170);
          myservo3.write(170);
        } else{
          aberto = 0;
          myservo1.write(0);
          myservo2.write(0);
          myservo3.write(0);
        }
        contagem = 0;
    }
    tempo = millis();
  }
```

```
void abreFech(Servo * serv) {
  int i;

  for(i = 0;i <180;i++){
    (*serv).write(i);
    delay(15);
  }
  delay(3000);
  for(i = 179;i >0;i--){
    (*serv).write(i);
    delay(15);
  }
}
```

B. Parte 2 - Sensores de distância - Energia

```
#include <Wire.h>
#include "Font.h"
#include <string.h>
#include "images.h"
#include <Servo.h>

#define OLED_Write_Address 0x3C

Servo gServo;

const int trigPin1 = 18;
const int echoPin1 = 17;
```

```

const int trigPin2 = 15;
const int echoPin2 = 14;

long duration1;
char distanceCm1;
long duration2;
char distanceCm2;

int GarraAbre = 10;
int GarraFecha = 104;

/* Funcao para enviar dados para OLED */
void OLED_Data(char *DATA)
{
    int len = strlen(DATA);
    for (int g=0; g<len; g++)
    {
        for (int index=0; index<5; index++)
        {
/* Comeca a transmissao para o
dispositivo escravo */
        Wire.beginTransmission(OLED_Write_Address);
/* Data em fila para ser transmitida */
        Wire.write(0x40);
        Wire.write(ASCII[DATA[g] -
          0x20][index]);
/* Transmite os dados na fila e termina a
transmissao para o dispositivo
escravo */
        Wire.endTransmission();
    }
}

/* Funcao para enviar comandos para o
OLED */
void OLED_Command(char DATA)
{
    Wire.beginTransmission(OLED_Write_Address);
/* Data em fila para ser transmitida */
    Wire.write(0x00);
    Wire.write(DATA);
    Wire.endTransmission();
}

/* Funcao para limpar o OLED */
void OLED_clear(void)
{
/* Coluna do endereco inicial 0, Coluna
do endereco final 127, Pagina do
endereco inicial 0, Pagina do
endereco final 7 */
    OLED_setXY(0x00, 0x7F, 0x00, 0x07);
    for (int k=0; k<=1023; k++)
    {
        Wire.beginTransmission(OLED_Write_Address);
        /* Data em fila para ser transmitida */
        Wire.write(0x40);
        Wire.write(0x00);
        Wire.endTransmission();
    }
}

/* Funcao para as configuracoes dos
enderecos da tela OLED */
void OLED_setXY(char col_start, char
col_end, char page_start, char
page_end)
{
    Wire.beginTransmission(OLED_Write_Address);
    /* Data em fila para ser transmitida */
    Wire.write(0x00);
    Wire.write(0x21);
    Wire.write(col_start);
    Wire.write(col_end);
    Wire.write(0x22);
    Wire.write(page_start);
    Wire.write(page_end);
}

/* Funcao de inicializacao do OLED */
void OLED_init(void)
{
    OLED_Command(0xAE); /* Entire Display
OFF */
    OLED_Command(0xD5); /* Set Display
Clock Divide Ratio and Oscillator
Frequency */
    OLED_Command(0x80); /* Default Setting
for Display Clock Divide Ratio and
Oscillator Frequency that is
recommended */
    OLED_Command(0xA8); /* Set Multiplex
Ratio */
    OLED_Command(0x3F); /* 64 COM lines */
    OLED_Command(0xD3); /* Set display
offset */
    OLED_Command(0x00); /* 0 offset */
    OLED_Command(0x40); /* Set first line
as the start line of the display */
    OLED_Command(0x8D); /* Charge pump */
    OLED_Command(0x14); /* Enable charge
dump during display on */
    OLED_Command(0x20); /* Set memory
addressing mode */
    OLED_Command(0x00); /* Horizontal
addressing mode */
    OLED_Command(0xA1); /* Set segment
remap with column address 127 mapped
to segment 0 */
    OLED_Command(0xC8); /* Set com output
scan direction, scan from com63 to
com0 */
}

```

```

    com 0 */
OLED_Command(0xDA); /* Set com pins
    hardware configuration */
OLED_Command(0x12); /* Alternative com
    pin configuration, disable com
    left/right remap */
OLED_Command(0x81); /* Set contrast
    control */
OLED_Command(0x80); /* Set Contrast to
    128 */
OLED_Command(0xD9); /* Set pre-charge
    period */
OLED_Command(0xF1); /* Phase 1 period
    of 15 DCLK, Phase 2 period of 1 DCLK
*/
OLED_Command(0xDB); /* Set Vcomh
    deselect level */
OLED_Command(0x20); /* Vcomh deselect
    level ~ 0.77 Vcc */
OLED_Command(0xA4); /* Entire display
    ON, resume to RAM content display */
OLED_Command(0xA6); /* Set Display in
    Normal Mode, 1 = ON, 0 = OFF */
OLED_Command(0x2E); /* Deactivate
    scroll */
OLED_Command(0xAF); /* Display on in
    normal mode */

}

/* Funcao para enviar imagens para o OLED
 */
void OLED_image(const unsigned char
    *image_data)
{
    OLED_setXY(0x00, 0x7F, 0x00, 0x07);
    for (int k=0; k<=1023; k++)
    {
        Wire.beginTransmission(OLED_Write_Address);
        /* Data em fila para ser transmitida */
        Wire.write(0x40); /* Para transmissao
            dos dados, C = 0 and D/C = 1 */
        Wire.write(image_data[k]);
        Wire.endTransmission(); /* Transmite
            os dados na fila e termina a
            transmissao para o dispositivo
            escravo */
    }
}

void setup() {
    Wire.begin(); /* Inicia a biblioteca
        Wire e seta I2C como master */
    OLED_init(); /* Inicia o OLED */
    pinMode(trigPin1, OUTPUT);
    pinMode(echoPin1, INPUT);
    pinMode(trigPin2, OUTPUT);
    pinMode(echoPin2, INPUT);

    pinMode(echoPin2, INPUT);
    gServo.attach(4); /* Seta o pino 4
        (P1_4) para o servo */
    delay(100);
    OLED_clear(); /* Clear OLED */
    /* Printa o Logo do Launchpad na tela
        OLED*/
    OLED_image(Launchpad_Logo);
    delay(1000);
    OLED_clear(); /*Limpa a tela de OLED*/
}

void Sensor_Ultral_Oled(void)
{
    OLED_setXY(0x00, 0x7F, 0x00, 0x03);
    /* Printa a string "Distance1" no OLED */
    OLED_Data("Distance1: ");
    OLED_setXY(0x36, 0x7F, 0x00, 0x03);
    /* Printa o valor da distancial como
        string no OLED (vai de 0 a 9) */
    OLED_Data(&distanceCm1);
    OLED_setXY(0x40, 0x7F, 0x00, 0x03);
    OLED_Data(" cm");

}

void Sensor_Ultra2_Oled(void)
{
    OLED_setXY(0x00, 0x7F, 0x03, 0x03);

    /*Printa a string "Distance2" no OLED*/
    OLED_Data("Distance2: ");
    /*Printa o valor da distancia2 como
        string no OLED (vai de 0 a 9)*/
    OLED_setXY(0x36, 0x7F, 0x03, 0x03);
    OLED_Data(&distanceCm2);
    OLED_setXY(0x40, 0x7F, 0x03, 0x03);
    OLED_Data(" cm");

    /*Calculo da distancia em centimetros com
        base na
        duracao dos pulsos dos pinos trigger e
        echo do sensor 1*/
    void loop() {

        digitalWrite(trigPin1, LOW);
        delayMicroseconds(2);
        digitalWrite(trigPin1, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin1, LOW);
        duration1 = pulseIn(echoPin1, HIGH);
        distanceCm1= duration1*0.034/2 + '0';

        /*Calculo da distancia em centimetros com
            base na
            duracao dos pulsos dos pinos trigger e
            echo do sensor 2*/
    }
}

```

```

digitalWrite(trigPin2, LOW);
delayMicroseconds(2);
digitalWrite(trigPin2, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin2, LOW);
duration2 = pulseIn(echoPin2, HIGH);
distanceCm2= duration2*0.034/2 + '0';

/*Funcao condicional para fechar a garra
quando a distancia
de um objeto do sensor 1 e 2 for igual e
menor ou igual que 3*/
if ((distanceCm2 - '0') ==
    (distanceCm1 - '0') &&
    (distanceCm2 - '0' <= 3)) {
    Sensor_Ultra1_Oled();
    Sensor_Ultra2_Oled();
    gServo.write(GarraFecha);
    delay(5000);
}
else if (distanceCm2 - '0' != 
    (distanceCm1 - '0') &&
    (distanceCm2 - '0' > 3));
{
    Sensor_Ultra1_Oled();
    Sensor_Ultra2_Oled();
    gServo.write(GarraAbre);
}
delay(10);
}

}

// Usar o leitor ADC para calculo da
// forca
float fsrV = fsrADC * VCC / 1023.0;
float fsrR = R_DIV * (VCC / fsrV -
1.0);
Serial.println("Resistance: " +
String(fsrR) + " ohms");
// Forca estimada com base na
// resistencia do sensor conforme
// FSR datasheet:
float force;
float fsrG = 1.0 / fsrR; // Calcula a
conductancia
if (fsrR <= 600)
    force = (fsrG - 0.00075) /
0.00000032639;
else
    force = fsrG / 0.000000642857;
Serial.println("Force: " +
String(force) + " g");
Serial.println();

delay(500);
}
else
{
}
}

```

C. Parte 3 - Sensor de força - Energia

```

//Pino conectado no FSR/divisor de tensao
const int FSR_PIN = 2;

const float VCC = 4.98; // Voltagem
medida na fonte de 5V utilizada
const float R_DIV = 997.0; // Resistencia
medida no resistor de 3,3K

void setup()
{
    Serial.begin(9600);
    pinMode(FSR_PIN, INPUT);
}

void loop()
{
    int fsrADC = analogRead(FSR_PIN);
    // Se FSR nao tem nenhuma pressao, a
    // resistencia sera
    // tendendo ao infinito. Entao a
    // voltagem sera proximo a 0.
    if (fsrADC != 0) // Se o leitor
    analogico for diferente de 0

```

APÊNDICE
CÓDIGO FINAL PARA O PC4 (PC3 COM
ALTERAÇÕES)

```
#include <msp430fr2433.h>
#include <stdio.h>
#include <stdlib.h>

//P1.x
#define TrigPin1 BIT7
#define EchoPin1 BIT6

//P2.x
#define TrigPin2 BIT5
#define EchoPin2 BIT6

//P1.1
#define PORTA1 BIT1

//P1.2
#define PORTA2 BIT2

//Ultrasensors
int miliseconds1, miliseconds2;
int distance1, distance2;
long sensor1, sensor2;

//Force Sensor
unsigned int ADC_Result;
const float VCC = 4.98; // Measured
    voltage of Arduino 5V line
const float R_DIV = 3220.0; // Measured
    resistance of 3.3k resistor

//Functions
void atraso(volatile unsigned int T_u);
void servo(volatile int graus);
void StopServo();
void movimento(volatile int graus);
void initUart();
void sendData(unsigned char c);
void sendInt(unsigned int n);
void sendString(char str[]);
void sendFloat(float f);
void Init_servos();
void Init_UltraSensors();
void Init_ForceSensor();
void ForceSensorMeasure(unsigned int
    ADC_Result);

void main(void)
{
    CSCTL1 = 0xF1; //MCLK e SMCLK @ 1MHz
    CSCTL0 = 0x00; //MCLK e SMCLK @ 1MHz
    WDTCTL = WDTPW + WDTHOLD; // Stop WDT
```

```
PM5CTL0 &= ~LOCKLPM5;

TA2CCTL0 = CCIE; // CCR0 interrupt
    enabled
TA3CCTL0 = CCIE;
TA2CCR0 = 1000; // 1ms at 1mhz
TA3CCR0 = 1000;
TA2CTL = TASSEL_2 + MC_1; // SMCLK,
    upmode
TA3CTL = TASSEL_2 + MC_1;

P1IFG = 0x00; //clear all interrupt
    flags
P2IFG = 0x00;
P1DIR |= 0x01;
P1OUT &= ~0x01;

//Configure Force Sensor
Init_ForceSensor();

__BIS_SR(GIE);

volatile int fechado = 0;
while(1)
{
    ADCCTL0 |= ADCENC | ADCSC; //
        Sampling and conversion start
    __bis_SR_register(GIE);

//Configure Serial UART
initUart();

ForceSensorMeasure(ADC_Result);
Init_UltraSensors();
atraso(1000);
if(fechado == 0){ //Mao aberta
    if(distance1<= 7 && distance2 <=
        4)
    {
        P1OUT |= 0x01;
        movimento(180);
        atraso(2000); //2seg
        P1OUT &= ~0x01;
        fechado = 1;
    }
    else if(distance1 > 7 &&
        distance2 > 4){
        StopServo();
    }
}else{ //Mao fechada
    if(distance1<= 7 && distance2 <=
        4){

        P1OUT |= 0x01;
        movimento(0);
        atraso(2000); //2seg
```

```

        P1OUT &= ~0x01;
        fechado = 0;
    }
    else if(distance1 > 7 &&
             distance2 > 4){
        StopServo();
    }
}
}

void atraso(volatile unsigned int T_u)
{
    TA1CCR0 = 999;
    TA1CTL = TACLR;
    TA1CTL = TASSEL_2 + ID_0 + MC_1;
    while(T_u>0)
    {
        while((TA1CTL&TAIFG)==0);
        T_u--;
        TA1CTL &= ~TAIFG;
    }
    TA1CTL = MC_0;
}

void servo(volatile int graus){
    if(graus <= 180 && graus >= 0)
        TA0CCR1 = 730 + 12*graus;
        TA0CCR2 = 730 + 12*graus;
}

void StopServo(){
    TA0CTL = MC_0;
}

volatile int posAnt=0;
void movimento(volatile int graus){
    volatile int i;
    Init_servos();
    TA0CTL = TACLR;
    TA0CTL = TASSEL_2 + MC_1; // SMCLK, up
    mode
    if(graus <= 180 && graus >= 0){
        if(posAnt > graus){
            for(i=posAnt;i >= graus; i=i-10){
                servo(i);
                atraso(90);
            }
        }
        else
            if(posAnt < graus){
                for(i=posAnt;i <= graus;
                    i=i+10){
                    servo(i);
                    atraso(90);
                }
            }
    }
}

void initUart() // 9600 Baudrate
{
    // Configure UART
    UCA0CTLW0 |= UCSWRST;
    UCA0CTLW0 |= UCSSEL__SMCLK;

    // Baud Rate calculation
    UCA0BR0 = 6;
    UCA0BR1 = 0x00;
    UCA0MCTLW = 0x20 | UCOS16 | UCBRF_8;

    UCA0CTLW0 &= ~UCSWRST;           // Initialize eUSCI
    UCA0IE |= UCRXIE;               // Enable USCI_A0 RX interrupt

    // Configure UART pins
    P1SEL0 |= BIT4 | BIT5;          // set
                                    // 2-UART pin as second function
}

void Init_servos(){
    P1DIR |= PORTA1 + PORTA2;

    P1OUT &= ~(PORTA1 + PORTA2);
    P1SEL1 |= PORTA1 + PORTA2; //Activate
                                // comp. mode. TA0.1 and TA0.2

    TA0CCR0 = 20000-1; // PWM Period TA0.x
    TA0CCR1 = 00; // CCR1 PWM duty cycle
    TA0CCR2 = 00; // CCR2 PWM duty cycle
    TA0CCTL1 = OUTMOD_7; // CCR1 reset/set
    TA0CCTL2 = OUTMOD_7; // CCR2 reset/set
}

void Init_UltraSensors(){

    P1SEL0 &= 0x00;
    P1SEL1 &= 0x00;
    P1IE &= ~0x01;
    P2IE &= ~0x01;
    P1DIR |= TrigPin1;
    P2DIR |= TrigPin2;
    P1OUT |= TrigPin1;
    P2OUT |= TrigPin2;
    __delay_cycles(10);
    P1OUT &= ~TrigPin1;
    P2OUT &= ~TrigPin2;
    P1DIR &= ~EchoPin1;
    P2DIR &= ~EchoPin2;
}

```

```

P1IFG = 0x00;
P2IFG = 0x00;
P1IE |= EchoPin1;
P2IE |= EchoPin2;
P1IES &= ~EchoPin1;
P2IES &= ~EchoPin2;

atraso(30);
distance1 = sensor1/74;
distance2 = sensor2/66;
}

void Init_ForceSensor(){
    // Configure ADC A3 pin
    SYSCFG2 |= ADCPCTL3;

    // Configure ADC10
    ADCCTL0 |= ADCSHT_4 | ADCON;
    ADCCTL1 |= ADCSHS_0 | ADCSHP |
        ADCSSEL_0;
    ADCCTL2 |= ADCRES;
    ADCMCTL0 |= ADCINCH_3 | ADCSREF_0;
    ADCIE |= ADCIE0;
}

void ForceSensorMeasure(unsigned int
ADC_Result){
if (ADC_Result != 0)
{
    float fsrV = ADC_Result * VCC /
        1023.0;

    float fsrR = R_DIV * (VCC / fsrV -
        1.0);

    //Envio via serial da resistencia
    // interna do sensor
    sendString("\b Resistencia: ");
    sendFloat(fsrR);
    sendString(" ohms\n");

    // Forca estimada com base na
    // resistencia do sensor conforme
    // FSR datasheet:

    float force;
    float fsrG = 1.0 / fsrR;
    if (fsrR <= 600)
        force = (fsrG - 0.00075) /
            0.00000032639;
    else
        force = fsrG / 0.000000642857;

    //Envio via serial da forca
    // calculada para o sensor em float
    sendString("Forca - float: ");
    sendFloat(force);
    sendString(" g\n");
}

//Envio via serial da forca
// calculada para o sensor em int
sendString("Forca - int: ");
sendInt((unsigned int)force);
sendString(" g\n");

//Envio via serial da forca
// calculada para o sensor em string
char str[12];
sprintf(str, "%d", (int)force);
sendString("Forca - String:");
sendString(str);
sendString(" g\n");

atraso(5);
}
else
{
}
}

void sendData(unsigned char c)
{
    while(!(UCA0IFG&UCTXIFG));
    UCA0TXBUF = c;
}

void sendInt(unsigned int n)
{
    int num[] = {10000,1000,100,10,1};
    int pos, posI = 0, cnt, i;

    if(n==0)
        sendData('0');
    else
    {
        for (i = 4; i >= 0; i--)
            posI = n >= num[i] ? i : posI;

        for (pos = posI; pos < 5; pos++)
        {
            cnt = 0;
            while (n >= num[pos]) {
                cnt++;
                n -= num[pos];
            }
            sendData(cnt+'0');
        }
    }
}

void sendString(char str[])
{
}

```

```

unsigned int i;
for(i=0; str[i]!='\0'; i++)
    sendData(str[i]);
}

void sendFloat(float f)
{
    int fInteger, fIntegerFract;
    float fFract;

    if(f < 0)
    {
        sendData('-');
        f = -1*f;
    }

    fInteger = (int) f;
    fFract = f -fInteger;
    fFract = 100*fFract;
    fIntegerFract = (int) fFract;
    sendInt((unsigned int) fInteger);
    sendData('.');
    sendInt((unsigned int) fIntegerFract);
}

//P1 interrupt for Ultrasensor 1 distance
//calculation
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{
    if(P1IFG&EchoPin1)
    {
        if(!(P1IES&EchoPin1))
        {
            TA2CTL|=TACLR;
            miliseconds1 = 0;
            P1IES |= EchoPin1;
        }
        else
        {
            sensor1 =
                (long)miliseconds1*1000 +
                (long)TA2R;
        }
        P1IFG &= ~EchoPin1;
    }
}

//P2 interrupt for Ultrasensor 2 distance
//calculation
#pragma vector=PORT2_VECTOR
__interrupt void Port_2(void)
{
    if(P2IFG&EchoPin2)
    {
        if(!(P2IES&EchoPin2))
        {
            TA3CTL|=TACLR;
            miliseconds2 = 0;
            P2IES |= EchoPin2;
        }
        else
        {
            sensor2 =
                (long)miliseconds2*1000 +
                (long)TA3R;
        }
        P2IFG &= ~EchoPin2;
    }
}

//TimerA2 interrupt service routine
#pragma vector=TIMER2_A0_VECTOR
__interrupt void Timer_A0(void)
{
    miliseconds1++;
}

// TimerA3 interrupt service routine
#pragma vector=TIMER3_A0_VECTOR
__interrupt void Timer_A1(void)
{
    miliseconds2++;
}

// ADC interrupt service routine
#if defined(__TI_COMPILER_VERSION__) ||
defined(__IAR_SYSTEMS_ICC__)
#pragma vector=ADC_VECTOR
__interrupt void ADC_ISR(void)
#elif defined(__GNUC__)
void __attribute__
((interrupt(ADC_VECTOR))) ADC_ISR
(void)
#else
#error Compiler not supported!
#endif
{
    switch(__even_in_range(ADCIV, ADCIV_ADCIFG))
    {
        case ADCIV_NONE:
            break;
        case ADCIV_ADCOVIFG:
            break;
        case ADCIV_ADCTOVIFG:
            break;
    }
}

```

```

    case ADCIV_ADCHIIFG:
        break;
    case ADCIV_ADCLOIFG:
        break;
    case ADCIV_ADCINIFG:
        break;
    case ADCIV_ADCIFG:
        ADC_Result = ADCMEM0;
        break;
    default:
        break;
}
}

// UART interrupt service routine
#if defined(__TI_COMPILER_VERSION__) ||
    defined(__IAR_SYSTEMS_ICC__)
#pragma vector=USCI_A0_VECTOR
__interrupt void USCI_A0_ISR(void)
#elif defined(__GNUC__)
void __attribute__
    ((interrupt(USCI_A0_VECTOR)))
    USCI_A0_ISR (void)
#else
#error Compiler not supported!
#endif
{
    switch (__even_in_range(UCA0IV,USCI_UART_UCTXCPTIFG) )
    {
        case USCI_NONE: break;
        case USCI_UART_UCRXIFG:
            while( !(UCA0IFG&UCTXIFG) );
            UCA0TXBUF = UCA0RXB舅;
            __no_operation();
            break;
        case USCI_UART_UCTXIFG: break;
        case USCI_UART_UCSTTIFG: break;
        case USCI_UART_UCTXCPTIFG: break;
        default: break;
    }
}

```
