CENTRO DE DESENVOLVIMENTO E TRANFERÊNCIA DE TECNOLOGIA ASSISTIVA

Relatório Técnico Anual

SensePro

Autor:

Gustavo Pasqua de Oliveira Celani

Dezembro/2016

SUMÁRIO

INTRODUÇÃO

Neste relatório será apresentada a contribuição individual do autor para o desenvolvimento de seu projeto de iniciação científica (Adaptação de um projeto de robô humanoide impresso em 3D em uma prótese sensoriada de membro superior).

1. DESENVOLVIMENTO TÉCNICO

1.1 Software de *feedback* do sensoriamento

Foi desenvolvido um *software* para mostrar as respostas dos sensores em tempo real devido a necessidade de acompanhar o comportamento dos sensores de pressão.

O *back-end* foi desenvolvido na linguagem *Wiring* devido a API de comunicação embarcada no ATmega328 (microcontrolador da plataforma de prototipagem Arduíno UNO® utilizado no controle do protótipo e captação dos sinais dos sensores). Já o *front-end* foi desenvolvido em *Java*, criando uma interface gráfica dinâmica e intuitiva ao usuário (Figura 1).

A IDE escolhida foi a *Processing3*® para o *front-end*, uma vez que ela já tem a API de comunicação *Java-Wiring* na qual torna-se possível criar um objeto em *Java* de Arduino que é capaz de transmitir os comandos em um protocolo que o microrontrolador consiga interpretar. No *back-end* foi utilizado a IDE Arduino para executar os comandos recebidos e enviar dados do microcontrolador para o *software* utilizando comunicação UART (USB).

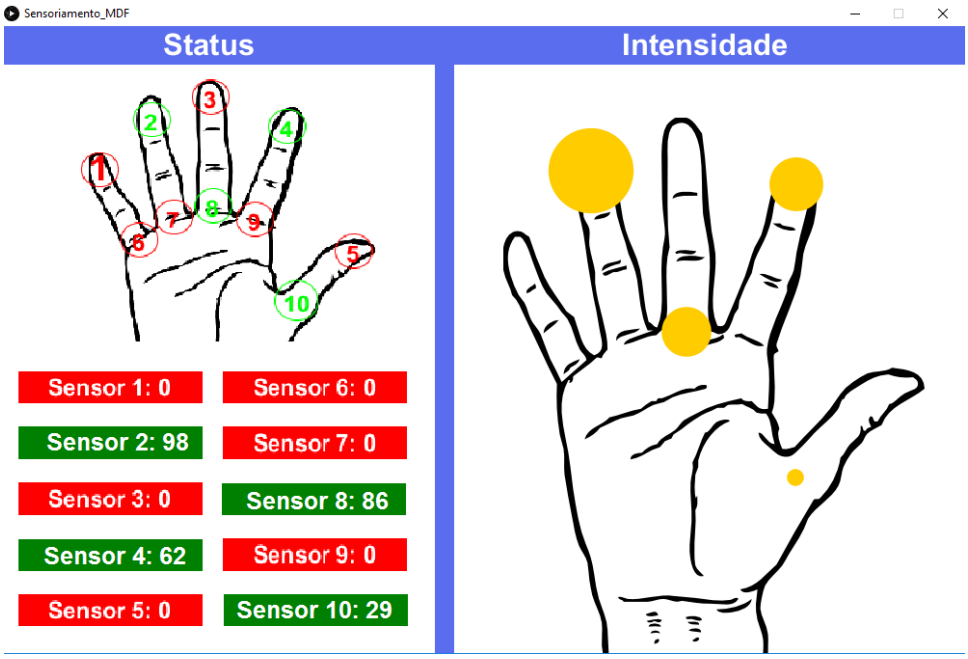

Figura 1 - *Front-end* do software de acompanhamento dos sensores

1.2 *Firmwares* para testes unitários

Foram desenvolvidos *firmwares* utilizando ideologia *plug&play* para testes rápidos de servo motores e de sensores pressão. A IDE utilizada foi a Arduino uma vez que os periféricos são controlados pelo ATmega328.

Para o teste dos servo motores, foi utilizada uma variação de tensão PWM (Pulse Width Modulation) na saída digital do microcontrolador que faz o motor movimentar-se, assim averiguando seu funcionamento.

No teste dos sensores de pressão, eles são ligados a entradas analógicas do microncontrolador com um circuito divisor de tensão, que lê e apresenta os valores de tensão captados já convertidos pelo conversor AD (Analogico-Digital) de 10bits presente no MCU. Com isso, pode-se aplicar pressão na área tátil do sensor e averiguar sua resposta analógica.

1.3 Algoritmo de fechamento de dedos simultâneos

O protótipo movimenta os dedos de forma independente, porém ao realizar o movimento de abrir e fechar a mão, ele movimentava um dedo de cada vez, isto criava uma movimentação lenta artificial. Uma solução seria criar um algoritmo capaz de realizar uma movimentação simultânea dos dedos.

Para movimentar os dedos simultaneamente seria necessário aplicar uma modulação de largura de pulso na tensão de saída dos 5 pinos digitais PWM que estão controlando os motores dos dedos no mesmo pulso de clock. Isto não é possível, pois a arquitetura RISC (Reduced Instruction Set Computer) presente no Arduino não permite este tipo de implementação.

Uma possível solução, seria reconfigurar os registradores do chip para que eles executem as 5 instrução necessárias em 5 pulsos de clock seguidos, isto daria um efeito visual de simultaneidade na movimentação dos dedos. Porém estes registradores não eram utilizados somente para movimentar os motores, e isto poderia prejudicar suas outras funções.

A solução utilizada foi implementar um algoritmo recursivo (Figura 2). Esta técnica de programação cria vários loops chamando a própria função dentro dela mesma passando novos parâmetros atualizados com a posição de cada dedo até que todos os loops forem resolvidos. Quando isso acontecer, o movimento estará completo.

```
void MOVIMENTO(int PINO_MOTOR, int POSICAO_ATUAL, int POSICAO_DESEJADA) {
  if(POSICAO_ATUAL > POSICAO_DESEJADA) POSICAO_ATUAL = POSICAO_ATUAL - TAXA;
  else if(POSICAO_ATUAL < POSICAO_DESEJADA) POSICAO_ATUAL = POSICAO_ATUAL + TAXA;
  else return;

  arduino.servoWrite(PINO_MOTOR, POSICAO_ATUAL);
  SELECIONA_PINO(PINO_MOTOR);
  MOVIMENTO(PINO_MOTOR, POSICAO_ATUAL, POSICAO_DESEJADA);
}
```

```
void SELECIONA_PINO(int PINO_MOTOR) {
  switch(PINO_MOTOR) {
    case PINO_DEDAO:
      ATL_DEDAO = POSICAO_ATUAL;
      PINO_MOTOR = PINO_INDICADOR;
      POSICAO_ATUAL = ATL_INDICADOR;
      POSICAO_DESEJADA = DSJ_INDICADOR;
      break;
    case PINO_INDICADOR:
      ATL_INDICADOR = POSICAO_ATUAL;
      PINO_MOTOR = PINO_MEDIO;
      POSICAO_ATUAL = ATL_MEDIO;
      POSICAO_DESEJADA = DSJ_MEDIO;
      break;
    case PINO_MEDIO:
      ATL_MEDIO = POSICAO_ATUAL;
      PINO_MOTOR = PINO_ANELAR;
      POSICAO_ATUAL = ATL_ANELAR;
      POSICAO_DESEJADA = DSJ_ANELAR;
      break;
    case PINO_ANELAR:
      ATL_ANELAR = POSICAO_ATUAL;
      PINO_MOTOR = PINO_MINIMO;
      POSICAO_ATUAL = ATL_MINIMO;
      POSICAO_DESEJADA = DSJ_MINIMO;
      break;
    case PINO_MINIMO:
      ATL_MINIMO = POSICAO_ATUAL;
      PINO_MOTOR = PINO_DEDAO;
      POSICAO_ATUAL = ATL_DEDAO;
      POSICAO_DESEJADA = DSJ_DEDAO;
      break;
  }
}
```

Figura 2 - Implementação recursiva de movimentação simultânea dos dedos

1.4 Redução do *hardware*

A fim de aumentar o espaço interno do protótipo para a adição do sensoriamento, foi necessário fazer uma redução do *hardware*. Para isso a placa dos motores foi reduzida com este propósito e foi confeccionada também a PCI dos sensores com os divisores de tensão. Os esquemas estão representados na Figura 3.

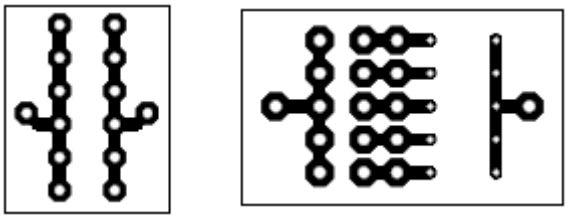

Figura 3 - PCI dos motores e PCI dos sensores

1.5 *Software* de controle mecânico e sensorial

Após a integração do sensoriamento na prótese, surgiu a necessidade de uma nova forma de controlá-la que fosse capaz de fazer um acompanhamento dos sensores ao mesmo tempo. Para tal função, foi desenvolvido este *software* que controla os movimentos do protótipo e atualiza os valores lidos dos sensores.

O *software*, da mesma forma que a versão anterior, foi desenvolvido utilizando a IDE *Processing 3*® devido aos fatores já comentados, com o front-end (Figura 4) em *Java* e o back-end em *Wiring* utilizando comunicação UART (USB).

Esta versão conta com os movimentos de abrir os dedos, fechar, rock, paz e fechamento parcial. O controle é feito por meio dos botões representando as posições do movimento. Existe um indicador que mostra a posição dos dedos juntamente com uma cor indicativa.

A cada segundo a aba "Sensores" atualiza os 5 sensores posicionados nos 5 dedos e apresenta seu valor em uma escala de 0 a 100 juntamente com uma barra de progresso. Também existe um indicador de carga que apresenta de forma geral, a carga que o protótipo está manuseando, sendo estes indicadores: "nenhuma carga", "carga baixa", "carga média", "carga alta", juntamente com uma cor indicativa para cada estado.

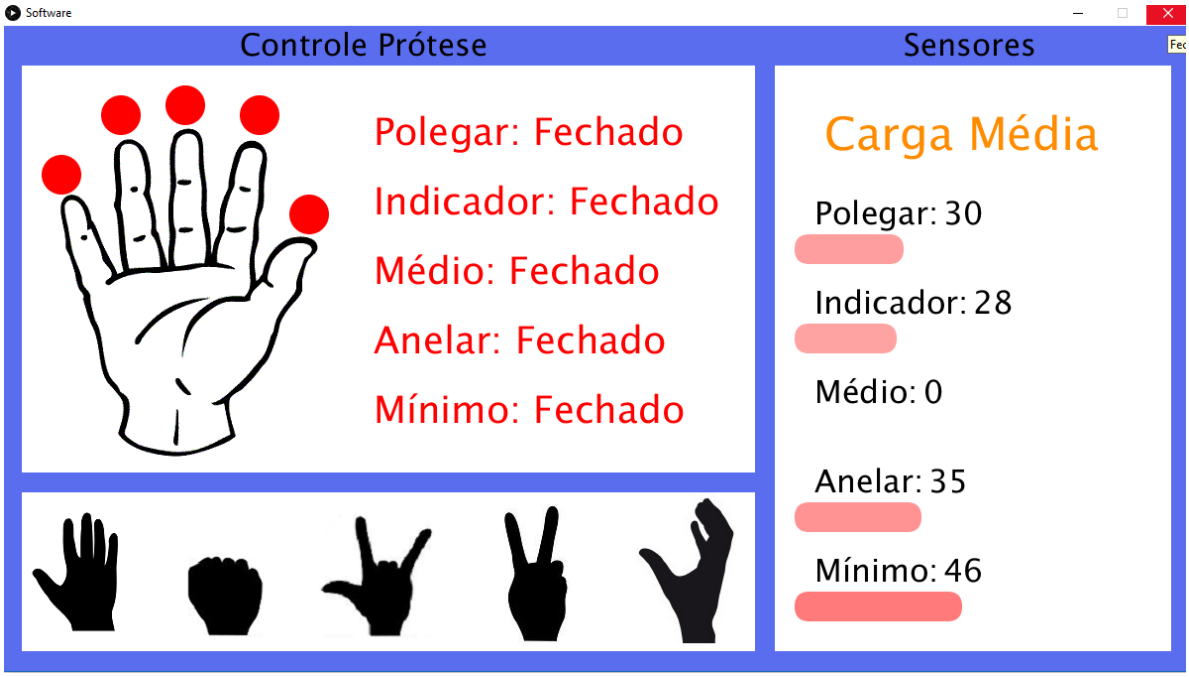

Figura 4 - Front-end do *software* de controle mecânico e sensorial

1.6 Teste comparativo entre os sensores de pressão

Os sensores de pressão piezoelétrico *MEAS*® não estavam respondendo adequadamente às massas. Consequêntemente foi realizado um teste comparando este modelo com um modelo de sensor resistivo da *InterLink Eletronics*® (FSR) para constatar qual seria mais adequado para esta aplicação.

Foi desenvolvido um *firmware* para a plataforma Arduino que apresenta os valores de ambos os modelos um ao lado do outro, podendo assim comparar as respostas obtidas no estímulo de cada sensor.

O teste concluiu que o sensor resistivo apresenta um comportamento logarítmico com área de precisão variável de acordo com a variação do circuito divisor de tensão. Enquanto o sensor piezoeletrico se comporta de forma digital, não sendo adequado para a proposta. O resultado é apresentado na Figura 5.

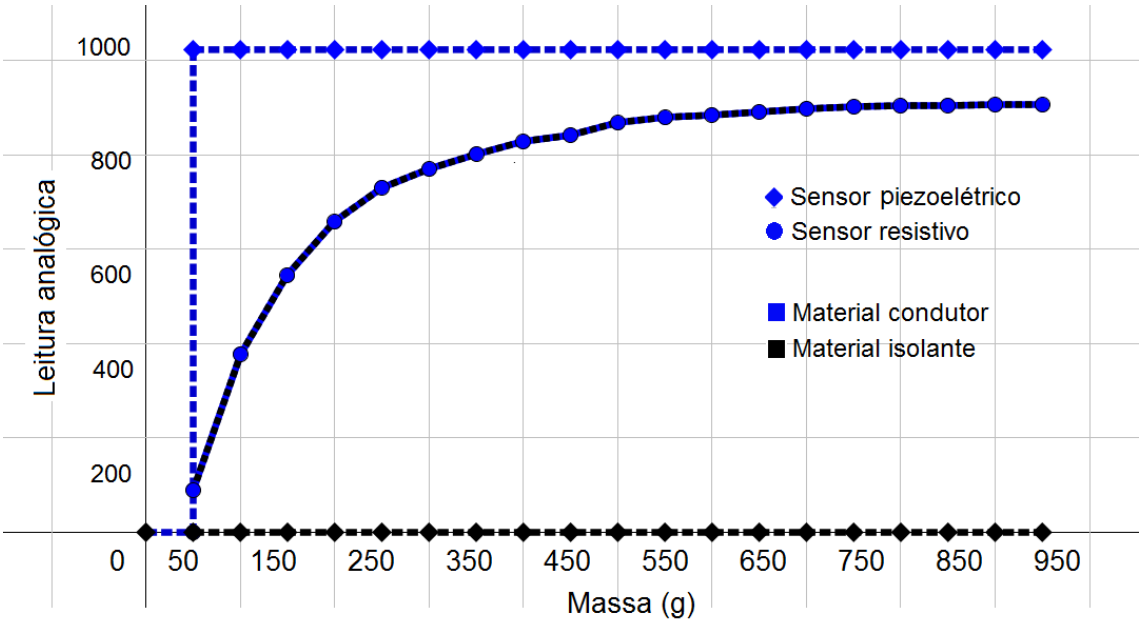

Figura 5 - Gráfico comparativo entre os sensores piezoelétrico e resistivo

1.7 Calibração automática dos sensores

Com os sensores fixados ao protótipo por meio de capinhas de silicone, eles adquiriam certo nível de tensão por estar em contato com o material, logo o valor inicial, que representa nenhuma massa em contato, não era idealmente zero. Para sanar este problema foi necessário fazer uma rotina de calibração para os sensores.

O algoritmo de calibração (Figura 6) foi implementado no *software* do protótipo e é executado apenas uma vez quando o *software* é iniciado. Ele gasta cerca de 3 segundos para calibrar os 5 sensores.

Assim que iniciada, a rotina aguarda 3 segundos para o estabelecimento dos sensores, lê os 5 valores obtidos dos 5 sensores, faz a conversão AD necessária para o reconhecimento e tranforma isso em uma taxa de *thereshold* para cada sensor. Logo cada sensor tem um *thereshold* diferente uma vez que ele depende de fatores externos.

Este algoritmo de calibração permite que o *software* apresente apenas informações reais do estado do sensoriamento. Uma vez que caso os sensores não passarem por esta função, a leitura dos valores serão realizadas de forma correta, porém não verídica com relação a carga manuseada pelo usuário.

```
void CALIBRAR() {

  delay(3000);

  SENSOR_DEDAO = arduino.analogRead(PINO_SENSOR_DEDAO);
  SENSOR_INDICADOR = arduino.analogRead(PINO_SENSOR_INDICADOR);
  SENSOR_MEDIO = arduino.analogRead(PINO_SENSOR_MEDIO);
  SENSOR_ANELAR = arduino.analogRead(PINO_SENSOR_ANELAR);
  SENSOR_MINIMO = arduino.analogRead(PINO_SENSOR_MINIMO);

  SENSOR_DEDAO = (int) map(SENSOR_DEDAO, 0, 1023, 0, 100);
  SENSOR_INDICADOR = (int) map(SENSOR_INDICADOR, 0, 1023, 0, 100);
  SENSOR_MEDIO = (int) map(SENSOR_MEDIO, 0, 1023, 0, 100);
  SENSOR_ANELAR = (int) map(SENSOR_ANELAR, 0, 1023, 0, 100);
  SENSOR_MINIMO = (int) map(SENSOR_MINIMO, 0, 1023, 0, 100);

  THERESHOLD_DEDAO = SENSOR_DEDAO;
  THERESHOLD_INDICADOR = SENSOR_INDICADOR;
  THERESHOLD_MEDIO = SENSOR_MEDIO;
  THERESHOLD_ANELAR = SENSOR_ANELAR;
  THERESHOLD_MINIMO = SENSOR_MINIMO;
}
```

Figura 6 - Algoritmo de calibração dos sensores

1.8 Teste de resistência do protótipo

Foi realizado um teste em quintuplicata de resistência para mensurar a massa suportada pelo protótipo. Nele foram adicionadas massas de 50g gradualmente até que o mesmo não consiga mais suportar o peso.

O teste foi bem sucedido, a média de carga máxima suportada pelo protótipo foi de 3,86Kg com desvio padrão de 0,22Kg e variância de 0,05Kg² com confiabilidade de 89,48% aferida pelo teorema de *Chebyshev*. A Tabela 1 apresenta os valores individuais de cada teste.

Teste	Massa trinc. mínimo (kg)	Tempo trinc. mínimo (h)	Carga máxima (kg)	Tempo total (h)
1	3,55	2:35:10	3,95	2:59:10
2	3,25	2:15:17	3,50	2:30:17
3	3,65	2:40:25	3,90	2:55:25
4	3,90	2:55:02	4,10	3:07:02
5	3,60	2:36:35	3,85	2:51:35

Tabela 1 - Resultados individuais do teste de resistência

1.9 Protocolo de comunicação bluetooth

Para o aprimoramento do protótipo, foi implementado uma comunicação bluetooth que realiza o tráfego de informações entre o dispositivo Android e o protótipo. Para isto, foi utilizado o módulo bluetooth HC-05 (Figura 7) que pode ter comportamento *master* ou *slave* (mestre ou escravo).

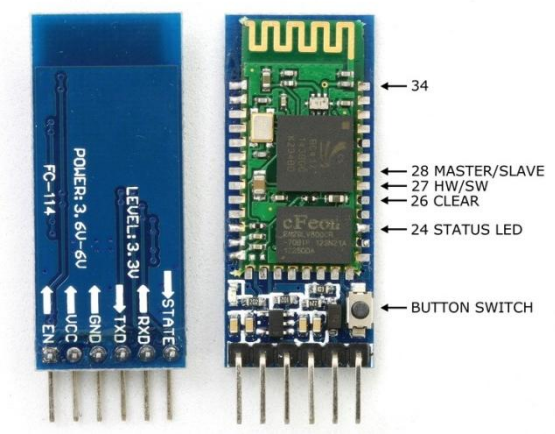

Figura 7 - Módulo bluetooth HC-05

Para esta aplicação, o módulo foi configurado para funcionar como *master* e o dispositivo Android como *device* (*slave*). A comunicação é estabelecida por meio do *serial number* do HC-05 que fica armazenado em memória *cache* na aplicação para facilitar os próximos pareamentos.

Uma vez pareado, o protocolo de comunicação é iniciado, nele existem duas funções principais: *SendData* e *ReceiveData*, sendo, respectivamente, uma função de envio de dados para o protótipo e outra de recebimento de dados do mesmo.

O protocolo implementado trabalha com o tráfego de 6 dígitos por ciclo. Sendo esses dígitos divididos em 3 segmentos de 2 dígitos cada. Estes dígitos são convertidos da tabela ASCII de caracteres para binário na hora da transmissão.

O primeiro segmento é responsável por identificar qual dedo está sendo aferido (Tabela 2), já os segundo e terceiro segmentos são responsáveis para transmitir o valor lido do sensor ou de posição do motor.

Dedo	String	ASCII
Polegar	a	97
Indicador	b	98
Médio	c	99
Anelar	d	100
Mínimo	e	101

Tabela 2 - Tabela de códigos dos dedos ASCII

O valor de posição do motor ou leitura analógica do sensor é um inteiro decimal de 2 dígitos que tem valor mínimo 10 e valor máximo 99. Isto faz com o que o comprimento da *String* a ser enviado seja sempre constante com 6 dígitos. Um exemplo do protocolo é apresentado na Figura 8.

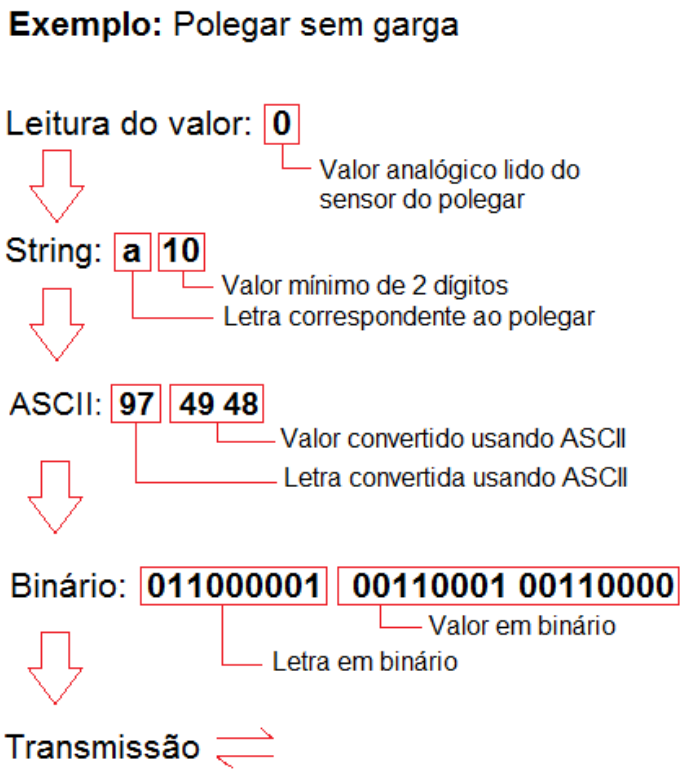

Figura 8 - Exemplo da aplicação do protocolo bluetooth implementado

1.10 Aplicação Android

Para fazer o gerenciamento do protótipo foi criado um aplicativo na plataforma Android. A versão 0.1 da aplicação foi desenvolvida utilizando *Java Android* na IDE *Android Studio*®. Esta versão contava apenas com o front-end sem nenhuma implementação de back-end. Esta etapa foi responsável por identificar os requisitos que o aplicativo precisaria atender e definir como seria uma interface mais funcional para a aplicação.

Depois de definido os requisitos, surgiu uma ideia de aprimoramento de comandos utilizando comandos de voz, para isto, a IDE *App Inventor*® atendeu melhor as necessidades do aplicativo que precisaria de comunicação bluetooth e reconhecimento de fala. Esta IDE tem uma API da *Google*® que gerencia sinais de áudio e o transformam em valores lógicos fazendo assim somente o tratamento de variáveis *booleanas* para comparar *strings* com sinais de áudio.

A aplicação, atualmente na versão 0.4.1 conta com funções de movimentos individuais dos dedos e um movimento sensoriado para manuseio de objetos controlando a intensidade de pressão que a prótese irá exercer na superfície. Este movimento sensoriado por ser comandado por meio de comandos de voz, o comando "fechar" faz com que a prótese feche seus dedos e interrompa o movimento assim que o sensor exercer a pressão pré-estabeleciada na superfície. Isto faz com que os dedos se moldem em torno do objeto sem danificá-lo. Já o comando "abrir" faz com que os 5 dedos se abram nas suas extensões máximas.

1.11 Algoritmo de sensibilidade

Para que o protótipo tivesse a funcionalidade de segurar objetos diversos exercendo uma pressão suficiente para agarrá-los porém sem danificá-los, foi implementado um algoritmo de sensibilidade.

O algoritmo (Figura 9) é capaz de fazer cada um dos 5 dedos, de forma independente, pararem em uma posição assim que o sensor nele acoplado identificasse que a pressão pré-selecionada no aplicativo foi alcançada. Além deste requisito, o algoritmo também é capaz de fazer este movimento com os 5 dedos de forma simultânea a olho nu.

Esta implementação foi realizada no *firmware* do protótipo na linguagem *Wiring*. Para o fechamento simultâneo foi utilizada a técnica de recursividade na função. Com o dedo inicialmente aberto o algoritmo move a uma taxa de 5 graus e aferi o valor do sensor, caso o valor esteja abaixo do valor de pressão pré-estabelecido, ele passa para o próximo dedo e repete o mesmo procedimento, até que os sensores dos 5 dedos estejam com a pressão próxima a programada pelo usuário. O procedimento é realizado recursivamente e cada dedo é analisado individualmente, garantindo uma empunhadura precisa do objeto.

```
void FechaSensoriado(int PINO_SENSOR, int THERESHOLD, int MOTOR_DEDO) {
  int VAL_SENSOR; //Valore de leitura do sensore
  int POS = 175;  //Inicialmente com o dedo aberto

  int INTENSIDADE = map(DataReceived[1], 10, 99, 5, 900);

  VAL_SENSOR = analogRead(PINO_SENSOR);
  VAL_SENSOR = map(VAL_SENSOR, THERESHOLD, 1023, 0, 1023);
  while(VAL_SENSOR <= INTENSIDADE && POS > TAXA) {
    POS = POS - TAXA;
    MOTOR_DEDO.write(POS);
    VAL_SENSOR = analogRead(PINO_SENSOR);
    VAL_SENSOR = map(VAL_SENSOR, THERESHOLD, 1023, 0, 1023);
  } POS = 175;

  SelecionaDedo(MOTOR_DEDO);
  FechaSensoriado(PINO_SENSOR, int THERESHOLD, int MOTOR_DEDO);
}
```

```
void SelecionaDedo(int MOTOR_DEDO) {
  switch(MOTOR_DEDO) {
    case MOTOR_POLEGAR:
      MOTOR_DEDO = MOTOR_INDICADOR;
      THERESHOLD = THERESHOLD_INDICADOR;
      PINO_SENSOR = PINO_SENSOR_INDICADOR;
      break;
    case MOTOR_INDICADOR:
      MOTOR_DEDO = MOTOR_MEDIO;
      THERESHOLD = THERESHOLD_MEDIO;
      PINO_SENSOR = PINO_SENSOR_MEDIO;
      break;
    case MOTOR_MEDIO:
      MOTOR_DEDO = MOTOR_ANELAR;
      THERESHOLD = THERESHOLD_ANELAR;
      PINO_SENSOR = PINO_SENSOR_ANELAR;
      break;
    case MOTOR_ANELAR:
      MOTOR_DEDO = MOTOR_MINIMO;
      THERESHOLD = THERESHOLD_MINIMO;
      PINO_SENSOR = PINO_SENSOR_MINIMO;
      break;
  }
}
```

Figura 9 - Algoritmo de sensibilidade

2. PARTICIPAÇÕES

2.1 INCITEL 2016

Participação e publicação no Congresso de Iniciação Científica do Inatel (INCITEL) com o projeto "OSMS" (Órtese sensorial de membro superior). Além do desenvolvimento técnico, foram envolvidos confecção de artigo científico e apresentação oral.

2.2 FETIN 2016

Participação e publicação na Feira Tecnológica do Inatel (FETIN) com o projeto "SensePro". Além do desenvolvimento técnico, foram envolvidos confecção de artigo para publicação no periódico impresso e apresentação do projeto no *stand* para o público. Nesta edição o projeto alcançou o quarto lugar do quarto nível do evento.

2.3 CBEB 2016

Participação e publicação no Congresso Brasileiro de Engenharia Biomédica (CBEB) com o projeto "Adaptação de um projeto de robô humanoide impresso em 3D em uma prótese sensoriada de membro superior". Além do desenvolvimento técnico, foram envolvidos confecção de artigo científico e elaboração de apresentação oral que foi realizada no congresso para o público.

2.4 Feiras acadêmicas

Participação em feiras acadêmicas pelo sul de Minas Gerais para divulgação do Inatel utilizando o projeto. Dentre as cidades visitadas, destacam-se: São José dos Campos, São Paulo e Americana.