UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO DE ELETRÔNICA
DEPARTAMENTO DE INFORMÁTICA
CURSO SUPERIOR DE ENGENHARIA DE COMPUTAÇÃO

TUI ALEXANDRE ONO BARANIUK

GARRA ROBÓTICA TELEOPERADA COM REALIMENTAÇÃO DE FORÇA

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2014

TUI ALEXANDRE ONO BARANIUK

GARRA ROBÓTICA TELEOPERADA COM REALIMENTAÇÃO DE FORÇA

Trabalho de Conclusão de Curso de Graduação, apresentado aos Departamentos de Eletrônica e Informática, da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do grau de Engenheiro de Computação.

Orientador: Prof. Dr. Hugo Vieira Neto

CURITIBA

2014

À minha mãe Maristela, ao meu pai James, à minha irmã Analin, à minha namorada Mariah e a toda minha família, por todo o apoio e carinho.

AGRADECIMENTOS

A minha mãe Maristela e ao meu pai James, pelo cuidado, pela ajuda, pelas ideias e incentivos.

Ao meu orientador, Prof. Dr. Hugo Vieira Neto, pela paciência, disponibilidade de tempo e material, indispensáveis para o desenvolvimento deste trabalho.

Aos meus amigos e colegas da faculdade, com os quais aprendi muito.

RESUMO

BARANIUK, Tui Alexandre Ono. **Garra robótica teleoperada com realimentação de força**. 2014. 60 f. Trabalho de Conclusão do Curso de Engenharia de Computação, Departamento de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

Para manipular objetos, a mão humana enfrenta dificuldades relacionadas a limitações físicas como tamanho do membro, tremores, fadiga e distância que alcança. Em uma cirurgia, robôs podem manipular instrumentos cirúrgicos muito menores e com maior precisão, podendo ser guiados por equipamentos de imagens médicas não invasivas como tomógrafos. Também podem operar em ambientes hostis ao ser humano, tais como grandes profundidades no oceano, trabalhar no espaço, dentre outras aplicações e possibilidades. Infelizmente, os sistemas disponíveis no mercado ainda não apresentam um retorno significativo da sensação de toque, de força e textura ao operador. Sendo assim, o objetivo do presente projeto é desenvolver um protótipo funcional composto por uma garra robótica de dois dedos teleoperada e um dispositivo a ser acoplado a uma mão humana com realimentação de força, que denominamos de manipulador. O projeto foi dividido em etapas, sendo as principais: estudo das tecnologias, desenvolvimento da placa base e *firmware*, desenvolvimento isolado dos blocos e posterior integração dos mesmos, assim como os respectivos testes. Foram utilizados servo motores e motores de passo para movimentação da garra e retorno de força ao usuário, assim como componentes mecânicos para construção das pinças. Para captar a força exercida pela garra e pela mão do operador foram utilizados sensores de força resistivos, denominados de FSR (*Force Sensor Resistor*) e *Micro Load Cells*. Para leitura e atuação nos componentes utilizou-se uma placa Arduino Uno, uma plataforma aberta e baseada no micromanipulador Atmega328. O protótipo é capaz de acompanhar o movimento de pinça do polegar e indicador do usuário, além de acompanhar a rotação do seu punho, criando uma resistência proporcional à leitura obtida pelos sensores da garra robótica, assim como este último se posiciona de acordo com a posição do manipulador. O projeto pode ser adaptado para outras aplicações práticas e, também, como plataforma para o ensino de robótica.

Palavras-chave: Garra Robótica. Teleoperação. Realimentação de Força. *Haptics* (sentido de tato).

ABSTRACT

BARANIUK, Tui Alexandre Ono. **Garra robótica teleoperada com realimentação de força**. 2014. 60 f. Trabalho de Conclusão do Curso de Engenharia de Computação, Departamento de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

In order to manipulate objects, the human hand faces problems related to physical limitations: arm length, tremors, fatigue and maximum reach. In a surgery robots can manipulate small surgical instruments with high precision, while guided by visual non-invasive equipments like tomographs. They can also operate in hostile ambients, such as depths in the ocean, space, amongst other applications and possibilities. Unfortunately, the systems available on the market still haven't shown significant results concerning response to touch sensation, force, and texture to the operator. Therefore, this project aims at developing a working prototype composed by a two-fingered teleoperated claw and a device attached to the human hand that is able to provide force feedback, so-called controller. The project has been divided in stages, being the main ones: study of technologies, development of the main base, firmware, and development of isolated blocks to be integrated and tested during the project. Servo and stepper motors were used to move de claw, and give force feedback to the user, as well as mechanical components were used for the construction of the tweezers. Force sensor resistors and micro load cells were used for reading the force excerced by the claw and the operator hand. The Arduino Uno board was used for controlling the system, which is an open-source plataform based on Atmega328 microcontroller. The prototype is able to follow the movement of the thumb and forefinger of the user, as well as to monitor the rotation of the wrist, creating resistence proportional to the readings obtained by the teleoperated claw sensors, as the latter is positioned according to the position of the controller. This project can be adapted to other pratical applications, and also as a plataform for teaching robotics.

Keywors: Robotic Claw. Teleoperation. Force Feedback. Haptics.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

FSR	*Force Sensor Resistor*
MLC	*Micro Load Cell*
AmpOp.	Amplificador Operacional

SUMÁRIO

1 INTRODUÇÃO

Para interagir com o ambiente, o ser humano utiliza como ferramenta seus diversos sentidos. O toque, diferente da visão ou audição, exige contato físico direto com o objeto (FISHEL, 2012). Com o sentido do tato podemos analisar textura, resistência e consistência, de modo que a utilização de tecnologias *haptics* é atrativa por permitir que usuários de computador manipulem objetos remotos ou virtuais com realimentação de força, mesmo não existindo contato direto entre ambos.

O estudo e desenvolvimento de plataformas que envolvem a sensação de toque envolvem múltiplas disciplinas: computação, engenharia, controle, robótica e controle motor humano (ZADEH, 2010). Aplicações podem ser encontradas na área de membros artificiais (próteses), tais como braços, garras, pernas, dentre outros. Na área médica, os membros artificiais podem substituir membros perdidos (ADEE, 2010), e exoesqueletos (CHIU, 2010) podem aumentar a força dos usuários. Também há grandes possibilidades de uso na área de máquinas controladas remotamente, como micro pinças que auxiliam em cirurgias que exigem extrema precisão (TAVAKOLI et al., 2007) e braços maiores e mais fortes para manipulação de objetos grandes e pesados (WEBSTER, 2010). Esses são alguns exemplos de tecnologias e projetos atuais.

Neste trabalho de conclusão de curso, foi construída uma garra robótica simplificada, controlada remotamente pelo usuário, por um dispositivo acoplado à sua mão, e dotado de atuadores e sensores, que neste trabalho denominamos de manipulador. O objetivo foi estender a possibilidade de manipulação de objetos por parte do usuário, sendo que este, ao utilizar um dispositivo que capta seus movimentos, transmite informações para a movimentação do protótipo em tempo real. O membro artificial também é capaz de medir a força aplicada sobre os objetos que manipula, sendo possível aplicar uma força proporcional diretamente à mão do usuário por meio do manipulador, transmitindo uma sensação de resistência e pressão, auxiliando na manipulação de objetos.

O projeto envolve tanto módulos físicos (*hardware*) quanto lógicos (*software*), que serão descritos detalhadamente neste documento. Serão descritas também as etapas de teste e validação do sistema proposto. Em seguida, serão discutidas ideias de extensão do projeto.

1.1 MOTIVAÇÃO

O uso de robôs em diversas aplicações e projetos voltados à manipulação de objetos tem crescido cada vez mais, sendo encontradas opções atraentes, devido à sua estabilidade e precisão (TAVAKOLI et al., 2007). Sua utilização em conjunto com computadores auxilia na superação de diversas dificuldades encontradas na manipulação de objetos: a movimentação da mão pode ser reduzida em escala para se obter maior precisão, tremores podem ser filtrados, o operador pode realizar o trabalho à longa distância e dentro de um ambiente confortável para reduzir a fadiga, dentre outros exemplos e possibilidades.

Estudos mostram que a utilização de retorno de força em atividades envolvendo manipulação de objetos melhora o desempenho e aumenta a eficiência das tarefas, reduzindo a força de pico causada pelo contato e, por consequência, reduzindo o consumo de energia (TAVAKOLI et al., 2007). O retorno de força também auxilia na manipulação de objetos virtuais (ROBLES, 2009), existindo outros estudos que mostram a importância do retorno de sensações táteis, que reduz o tempo para completar tarefas e melhora a percepção e capacidade motora do operador humano (TAVAKOLI et al., 2007).

A ausência de retorno de força na manipulação de objetos remotos traz dificuldades tais como a dependência visual para determinar a força do aperto da garra, problemas para analisar a consistência do objeto, dentre outros fatores, o que prolonga e dificulta a execução da tarefa.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver um protótipo funcional de uma garra robótica teleoperada por uma mão humana e com realimentação de força.

1.2.2 Objetivos Específicos

a) Construir o manipulador e o mecanismo da garra robótica;

b) Implementar realimentação de força e posição;
c) Implementar *firmware*.
d) Testar o sistema desenvolvido.

1.3 ESTRUTURA DO TRABALHO

O restante do presente trabalho está estruturado da seguinte forma:

No capítulo 2, são explicados os principais conceitos que embasam o projeto.

O capítulo 3 apresenta o diagrama geral do projeto e os componentes utilizados, além da metodologia de desenvolvimento empregada.

Por fim, são apresentados os resultados obtidos no capítulo 4 e as considerações finais no capítulo 5.

2 FUNDAMENTAÇÃO TEÓRICA

Para desenvolver o presente trabalho, outros projetos foram estudados, analisando-se as possíveis formas de abordar o problema. Para atingir os objetivos propostos, foi necessária a análise e escolha dos componentes a serem adquiridos, formas de captar a movimentação do usuário, como fazer o retorno da força, comunicação entre blocos e alimentação do conjunto. Na escolha dos componentes, os seguintes requisitos foram analisados: especificação, desempenho e valor. Para a pesquisa dos dispositivos, utilizaram-se como referência os elementos apresentados na Figura 1.

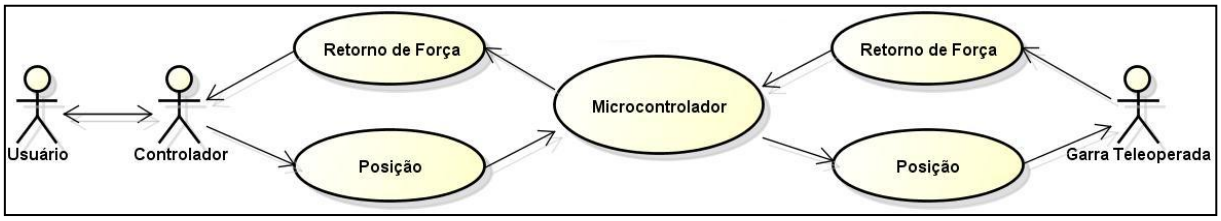


Figura 1 – Diagrama de casos de uso
Fonte: Autoria própria.

A garra robótica requer motores para sua movimentação e sensores para captar a força exercida sobre os objetos manipulados. Servo motores são muito comuns em aplicações na área de robótica, pela sua facilidade de uso, eficiência no consumo de energia (proporcional ao peso carregado), além de serem encontrados em dimensões reduzidas, sendo capazes de gerar um torque alto em relação ao seu tamanho. Inicialmente, foram testados microservos TP SG90 e servos TP MG995, que podem ser encontrados por 4 a 6 e 8 a 10 dólares, respectivamente, tornando-os opções atrativas para serem utilizados em protótipos e testes. Nos protótipos finais, estes servos foram substituídos por servos Futaba S3001, que são mais robustos e podem ser adquiridos por aproximadamente 20 dólares.

Além dos servos, os motores de passo também são boas opções, por sua alta precisão e torque. Para este projeto foram adquiridas duas unidades do modelo SM1.8-B2SB-SE, da Action Technology, um motor de passo unipolar com precisão de 1,8º por passo e capaz de gerar um torque de até 5,0 kgf/cm (ACTION TECHNOLOGY, 2010).

Para captar variações de pressão foram escolhidos sensores do tipo *FSR* (INTERLINK ELETRONICS, 2010) e *Micro Load Cell* (0–5Kg) (PHIDGETS, 2011),

disponíveis na faixa de 5 a 7 dólares. Os sensores *FSR* são compostos por duas membranas externas separadas por uma fina camada de ar que, quando são aproximadas, apresentam uma variação na resistência do sensor. Estes sensores são finos e leves, podendo ser equipados na garra robótica sem que sua presença atrapalhe no funcionamento do conjunto. Já o *Micro Load Cell* é uma pequena armação de metal que possui medidores de tensão em pontos específicos de sua estrutura, produzindo sinais elétricos proporcionais à carga que recebe.

A placa Arduino Uno pode ser encontrada por 30 dólares e possui os requisitos necessários ao projeto: é capaz de controlar múltiplos servos e ler os sensores por meio de pinos I/O e entradas analógicas, respectivamente. Sua plataforma é aberta e possui documentação e bibliotecas prontas disponíveis gratuitamente na internet (MARGOLIS, 2011).

Para concretizar o projeto, foi necessária uma análise a respeito das formas de movimentação da garra e retorno de força, assim como quais materiais poderiam ser utilizados para a montagem do protótipo.

2.1 *HAPTICS*

Interfaces *haptics* são dispositivos motores controlados via computador, que interagem fisicamente com operadores humanos para simular a presença de ambientes/objetos, utilizando o computador como mediador (PATOGLU; SATICI, 2010).

Para desenvolver uma interface *haptics* devem ser levados em conta: a velocidade de movimentação, a força e a precisão desejada, de acordo com a aplicação. A grande maioria dos dispositivos *haptics* busca trocar informações com o operador humano, substituindo *displays* tradicionais por interfaces táteis capazes de produzir força (BUERGER; HOGAN, 2010). Um dispositivo *hapitcs* que interage fisicamente com pessoas deve ser capaz de se mover e gerar uma força controlada, baseando-se na capacidade dos membros com os quais interage, e deve poder ser movido por forças muito menores que a capacidade máxima destes membros.

2.2 CONEXÃO

Tendo em vista manter o contato das pinças utilizadas pelo manipulador robótico com a mão do usuário, foi empregado um conceito de conexão baseado na invasão/ausência de corpos. Pode-se dizer que existe uma ligação entre dois corpos quando um é afetado pela movimentação do outro, de forma que qualquer alteração de posição seja notada. Para simular isso em dois corpos, é necessário existir inicialmente uma pressão mínima entre ambos. Quando uma ausência de pressão é detectada, sabe-se da ocorrência de perda de contato. Se um aumento de pressão ou força é detectado, está ocorrendo invasão de espaço.

Para implementar a ideia em um protótipo, os valores de saída dos sensores do manipulador são utilizados para verificar a movimentação dos corpos com relação ao aumento (invasão) ou diminuição (ausência) da força medida, com as garras se movendo de acordo [Figura 2]. Se o valor dado pelo sensor *FSR* estiver entre os limites de invasão e ausência, o sistema permanece em repouso. Se a leitura é menor que o limite inferior, referente à ausência de contato, o servo correspondente abre até que a resistência mínima seja alcançada. Caso a força aplicada seja superior ao limite de invasão, o servo é fechado, cedendo espaço.

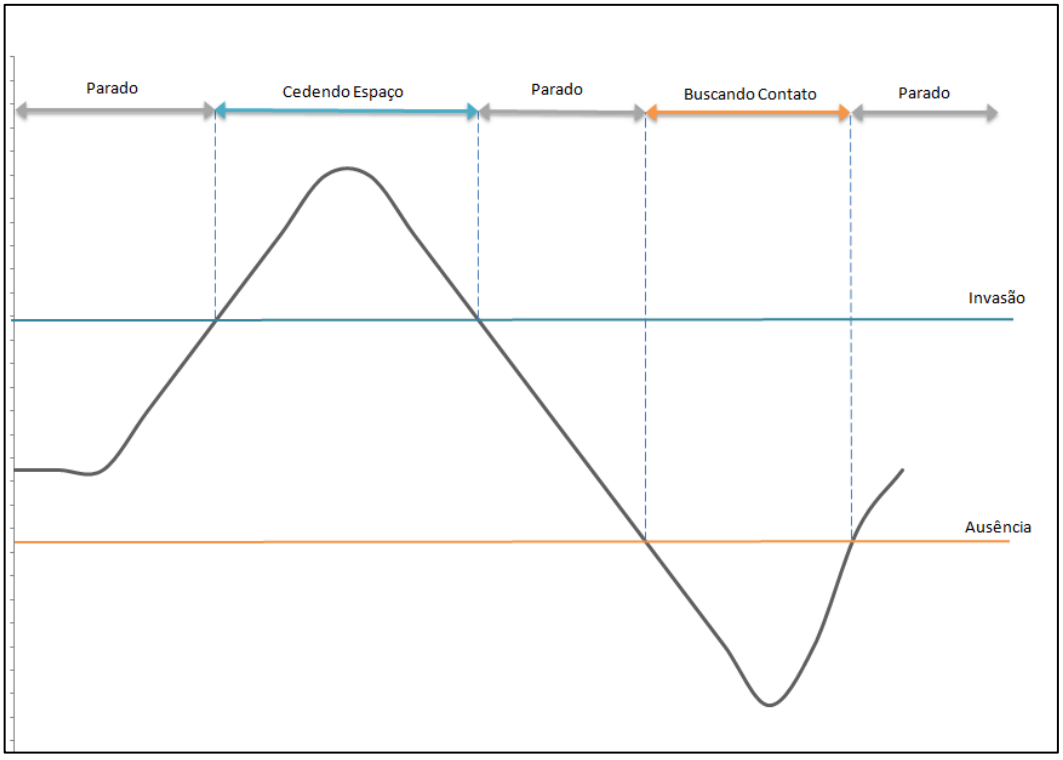


Figura 2 – Limites invasão ausência (Força x Tempo)
Fonte: Autoria própria.

3 METODOLOGIA

3.1 DIAGRAMA DO PROJETO

O sistema desenvolvido neste projeto final de curso, cujo diagrama geral é dado pela Figura 3, é composto basicamente por uma parte física (*Hardware*) e outra lógica (*Firmware*). Nas seções seguintes deste capitulo, serão descritos e detalhados os blocos que compõem o sistema apresentado, assim como cada componente escolhido.

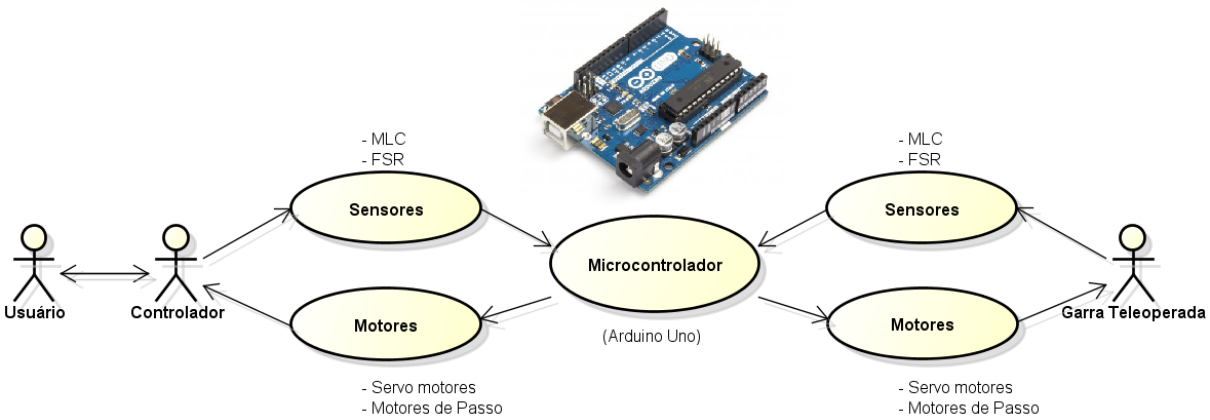


Figura 3 – Diagrama de componentes
Fonte: Autoria própria.

3.2 COMPONENTES

3.2.1 Arduino

O Arduino Uno é um sistema baseado no micromanipulador ATega328 (MARGOLIS, 2011), possui 14 pinos de I/O, 6 entradas analógicas com 10 bits de resolução, *clock* de 16Mhz, 3 *timers*, e pode ser alimentado via USB ou fonte externa. A Figura 4 mostra a visão de topo da placa utilizada no projeto.

Figura 4 – Placa Arduino Uno
Fonte: Autoria própria.

Neste projeto, foram utilizados 3 pinos referentes à alimentação:

- VIN: Tensão de entrada via fonte externa. Foi utilizado para alimentar os servo motores e pontes H;
- 5V: Gera uma saída com 5V de tensão por meio do regulador da placa. Foi utilizado na alimentação dos sensores *FSR* e *MLC*;
- GND: Pino de terra, utilizado tanto pelos sensores quanto pelos motores.

Para o controle dos motores, foram utilizados pinos de saída digital oferecidos pela placa. Uma saída digital foi reservada para cada servo presente no projeto, responsável por gerar sinal PWM. O PWM consiste num sinal digital de ondas retangulares de frequência constante, cuja fração de tempo, na qual o sinal permanece em nível alto (*duty cycle*), varia de 0 a 100%. Aproveitou-se uma biblioteca aberta, denominada Servo.h, que recebe como entrada o pino onde está conectada a entrada do servo e o *duty cycle* a ser utilizado pelo sinal de saída. Cada motor de passo ocupa quatro pinos de saída digital, que realizam o chaveamento sequenciado das bobinas responsáveis por gerar o movimento do conjunto.

Os sensores *FSR* e *MLC* são lidos pelas entradas analógicas da placa, por meio do mapeamento das tensões de entrada que variam de 0 a 5V para um valor inteiro entre 0 e 1023, ou seja, uma resolução de 0,0049mV por unidade. A placa leva aproximadamente 100µs para ler uma entrada analógica, velocidade suficiente para o funcionamento do protótipo, cujos motores são atualizados na casa dos milissegundos.

3.2.2 Sensores de força

3.2.2.1 *Force Sensor Resistor* (*FSR*)

Para captar a pressão exercida pelos dedos do usuário e pela garra robótica, foram utilizados sensores circulares Interlink com 5mm de diâmetro [Figura 5], que apresentam uma diminuição de resistência conforme uma força é aplicada sobre sua superfície ativa. Estes componentes são formados por duas membranas separadas por uma fina camada de ar, mantida ali por um adesivo ao redor das bordas. Uma das membranas é formada por duas trilhas eletricamente distintas, e a segunda por tinta FSR. Quando o sensor é pressionado, essas membranas se aproximam, e a tinta FSR conecta as trilhas com uma resistência proporcional à força aplicada (INTERLINK ELETRONICS, 2010).

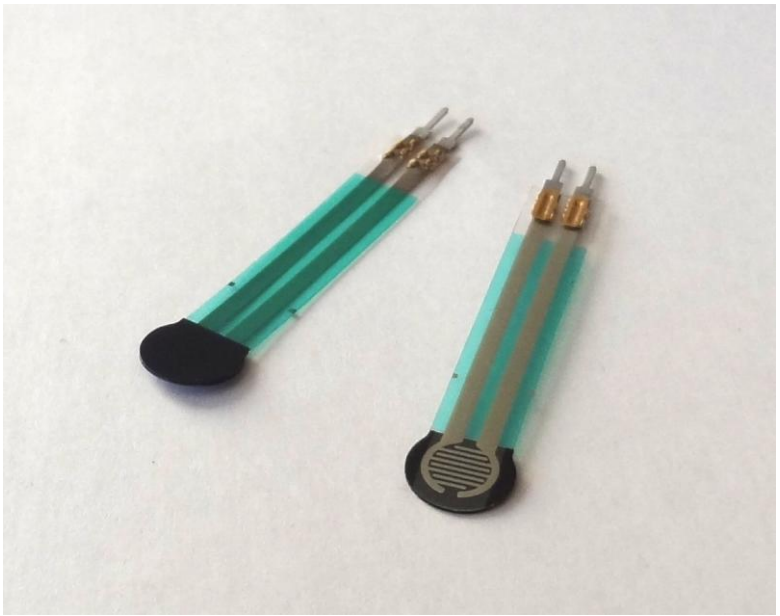


Figura 5 – Sensores *FSR* (frente e verso)
Fonte: Autoria própria.

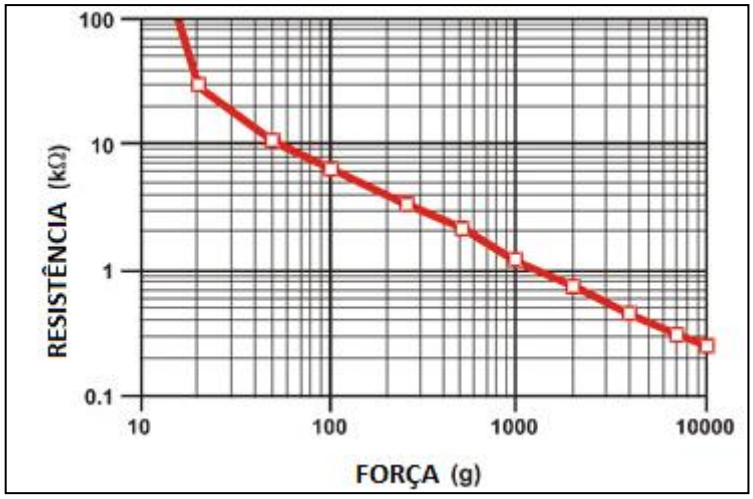


Figura 6 – Relação de Resistência x Força do sensor *FSR*
Fonte: Adaptado de Interlink, (2014, p. 5).

A Figura 6 apresenta a relação entre variação da resistência conforme a força aplicada. A calibragem dos sensores foi feita por meio da medição da resposta após a aplicação de pesos conhecidos, cujos dados são utilizados na seguinte função de mapeamento:

$$f = (x - in_{min}) \times \frac{(out_{max} - out_{min})}{(in_{max} - in_{min})} + out_{min}$$

Sendo: x o valor lido pela entrada analógica, in_{min} o valor analógico lido para um peso conhecido A, out_{min}, e in_{max} o valor analógico lido para um peso conhecido B, out_{max}.

A leitura do *FSR* pode ser feita pelo próprio Arduino, assim como a alimentação. Na Figura 7, é apresentado o circuito utilizado para um sensor *FSR*.

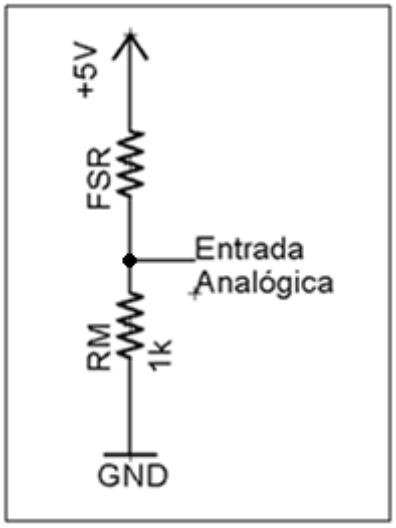


Figura 7 – Circuito divisor de tensão para o sensor *FSR*
Fonte: Autoria própria.

Pode-se observar que foi ligado um resistor Rm em série ao *FSR*, para compor um divisor de tensão: 5V são distribuídos pelos dois resistores, de modo proporcional à sua resistência. A resistência total do conjunto varia de 100kΩ a 1kΩ para forças aplicadas de 0 a 100N. Os sensores foram ligados às entradas analógicas 0 a 3 da placa, cada uma lê a tensão sob os respectivos resistores divisores de tensão, respondendo com um valor entre 0 e 1023, proporcional à leitura de 0 a 5V, conforme varia a resistência do *FSR*. Se não existe força sobre o sensor, a resistência deste será alta, e ele absorverá a maior parte da tensão do conjunto, de modo que a leitura, baseada na tensão sobre o resistor em série com o *FSR*, será baixa. Se existe força aplicada, a resistência do sensor diminui, e a tensão sobre o resistor Rm aumenta, assim como o valor lido pelo Arduino. A equação referente à tensão lida é a seguinte: $Vout = Rm * V/(Rm + Rfsr)$, sendo Rm o divisor de tensão, V a tensão fornecida pelo Arduino, igual a +5V, e $Rfsr$ a resistência variável do sensor.

3.2.2.2 *Micro Load Cell*

Para captar a pressão exercida pela torção da garra, foram utilizados sensores do tipo *Micro Load Cell* modelo CZL635 (PHIDGETS, 2011), capazes de medir cargas de até 5Kg. Na Figura 8, é apresentada a imagem e o diagrama elétrico deste sensor de força, que possui uma estrutura metálica com extensômetros (NATIONAL INSTRUMENTS, 2014) distribuídos em pontos

específicos. Quando submetido a uma tensão ou força, o sensor gera impulsos elétricos proporcionais à carga recebida, que podem ser medidas após amplificação do sinal para leitura.

(a) Fotografia (b) Diagrama Elétrico

Figura 8 – Sensor *MLC*
Fonte: Adaptado a partir do datasheet Micro Load Cell – CZL635, (PHIDGETS, 2011).

Os sinais elétricos gerados pelo *Micro Load Cell* são muito baixos, (1,0±0,15mV), não sendo possível realizar uma leitura direta pelo Arduino. Deste modo, conforme mostrado na Figura 9, foi criado um amplificador de instrumentação, utilizando o circuito integrado OP07CP(TEXAS INSTRUMENTS, 1996), que é um amplificador de baixo nível de ruído e de baixo *off-set*.

A tensão de saída é dada pela seguinte equação:

$$Vo = \left(1 + \frac{20000}{R}\right) \times (Va - Vb)$$

Nesta equação, R é um resistor de referência, que pode ser adaptado de acordo com a saída desejada, e Va e Vb são as tensões de saída do sensor de força, cuja diferença deseja-se amplificar para leitura. Neste projeto, foi utilizado um resistor R de 33Ω, resultando num ganho de aproximadamente 600 vezes a diferença entre as tensões Va e Vb.

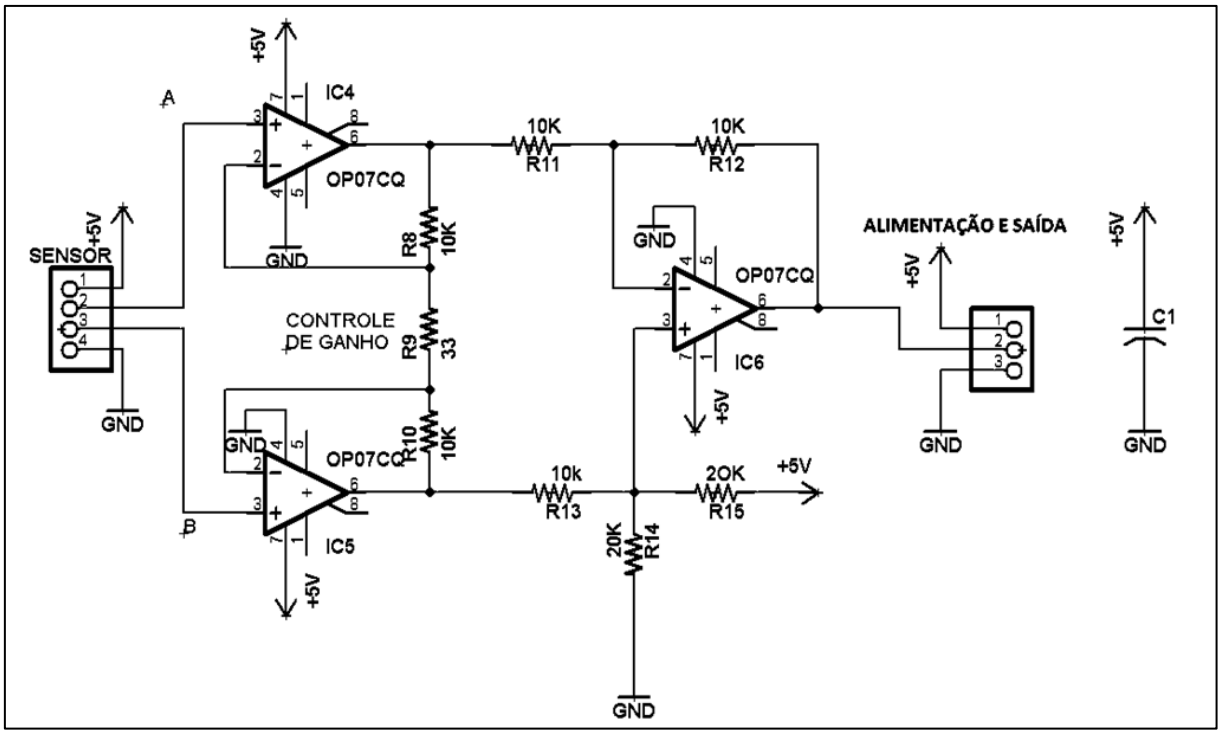


Figura 9 – Circuito amplificador de instrumentação

Fonte: Autoria própria.

3.2.3 Servo motores

No protótipo, foram utilizados inicialmente servos TP SG90 e MG995, mostrados na Figura 10. Por se tratar de um componente de baixo custo, foi possível a obtenção de componentes extras para teste. Posteriormente foram substituídos por servos da linha Futaba S3001, que, apesar do maior custo, apresentaram desempenho e robustez superiores aos primeiros.

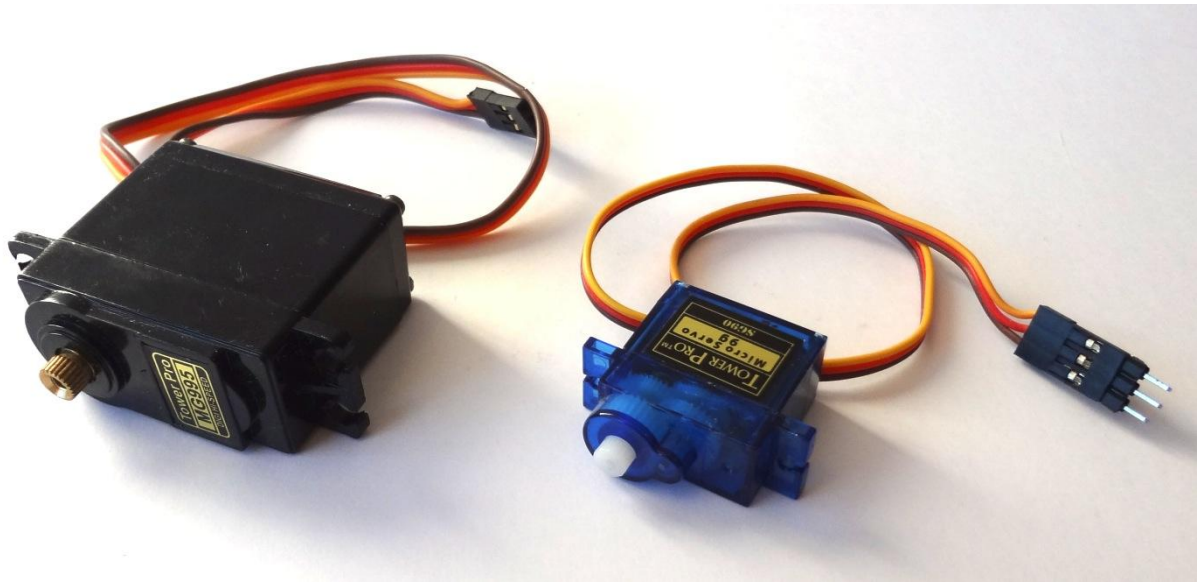


Figura 10 – Servos MG995 e TP SG90
Fonte: Autoria própria.

Os servos são alimentados por uma fonte externa de 6V e compartilham o *GND* com a placa Arduino, cujo micromanipulador é responsável por gerar os sinais de PWM e por definir as posições dos servos. O circuito eletrônico utilizado pelos servos é apresentado na Figura 11. A Figura 12 ilustra o funcionamento do PWM para alguns valores de *duty cycle* e a resposta dada pelos servos.

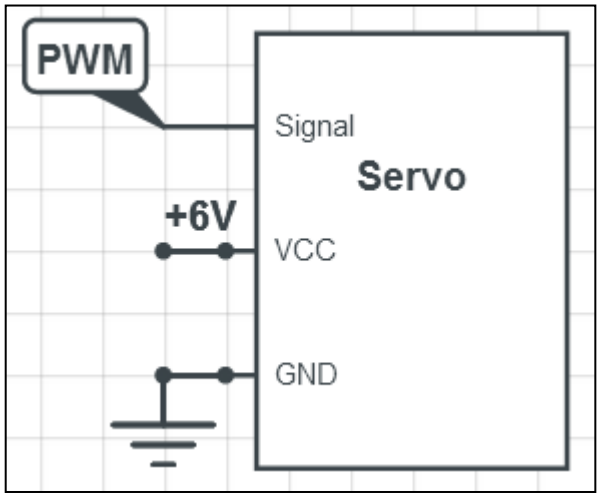


Figura 11 – Circuito dos servos
Fonte: Autoria própria.

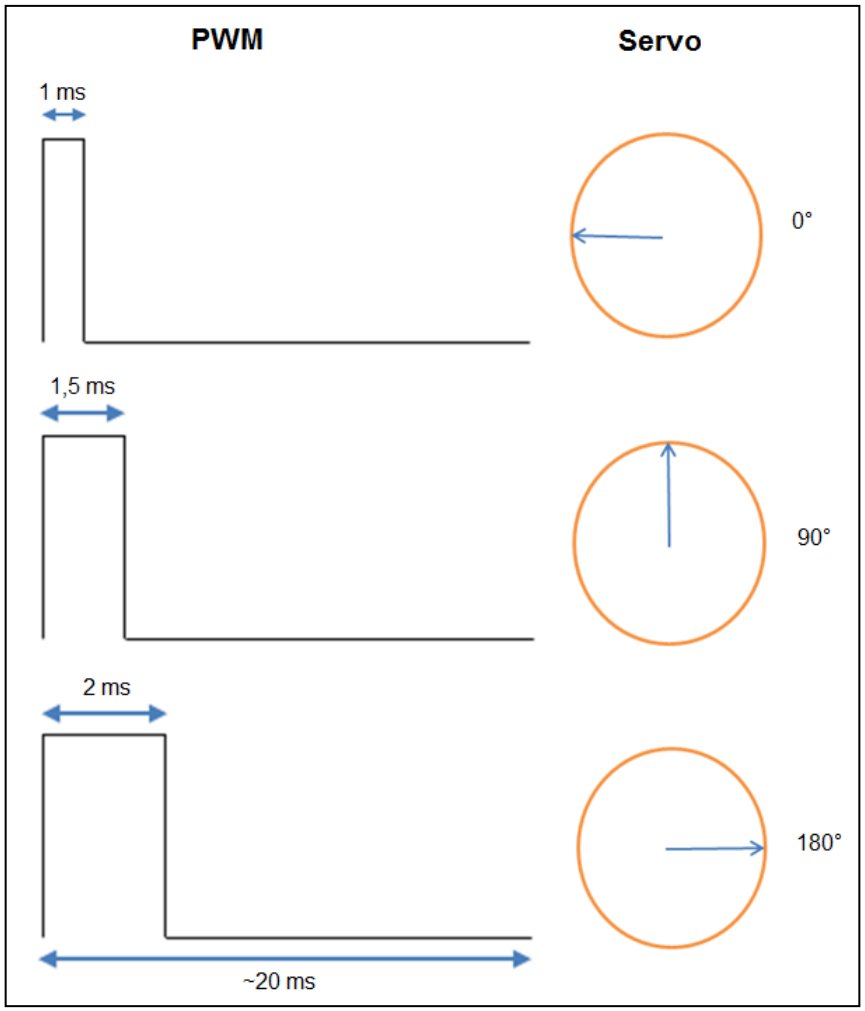


Figura 12 – Entrada PWM e posição do servo TP SG90
Fonte: Autoria própria.

Foram realizados testes de bancada para verificar a faixa de alcance do servo em relação à entrada PWM. O primeiro deles envolveu um servo isolado, e analisou-se a resposta do servo de acordo com a entrada PWM, cuja largura de onda variou-se de 1000 a 2000 microssegundos para o micro servo SG90 e para o servo MG995, e de 544 a 2350 microssegundos para o servo Futaba. Para testes com múltiplos servos, as entradas do sinal PWM dos servos foram conectadas cada uma a uma porta de saída digital do micromanipulador.

Para cada servo, verificou-se a resposta angular de acordo com o PWM de entrada, anotando-se a faixa de PWM utilizada para manter o servo dentro da faixa angular desejada. Isso possibilitou o remapeamento da faixa de *duty cycle* utilizada para manipular cada servo em específico, utilizando-se a mesma função da calibragem dos sensores de pressão:

$$f = (x - in_{min}) \times \frac{(out_{max} - out_{min})}{(in_{max} - in_{min})} + out_{min}$$

Sendo: x o ângulo desejado, in_{min} o ângulo medido para uma entrada PWM, out_{min}, e in_{max} o ângulo medido para uma entrada PWM, out_{max}. Os valores utilizados para out_{min} e out_{max} foram de 1000µs e 2000µs para o microservo TP e 554µs e 2350µs para o servo Futaba, respectivamente.

3.2.4 Motores de passo

Motores de passo funcionam a partir da energização sequenciada de duas ou mais bobinas independentes umas das outras, apresentando alta precisão e alto torque. Foram utilizados no projeto dois motores de passo unipolares SM1.8-B2SB-SE (ACTION TECHNOLOGY, 2010), com precisão de 1,8º por passo (200 passos por volta) e que funcionam com 12V de alimentação.

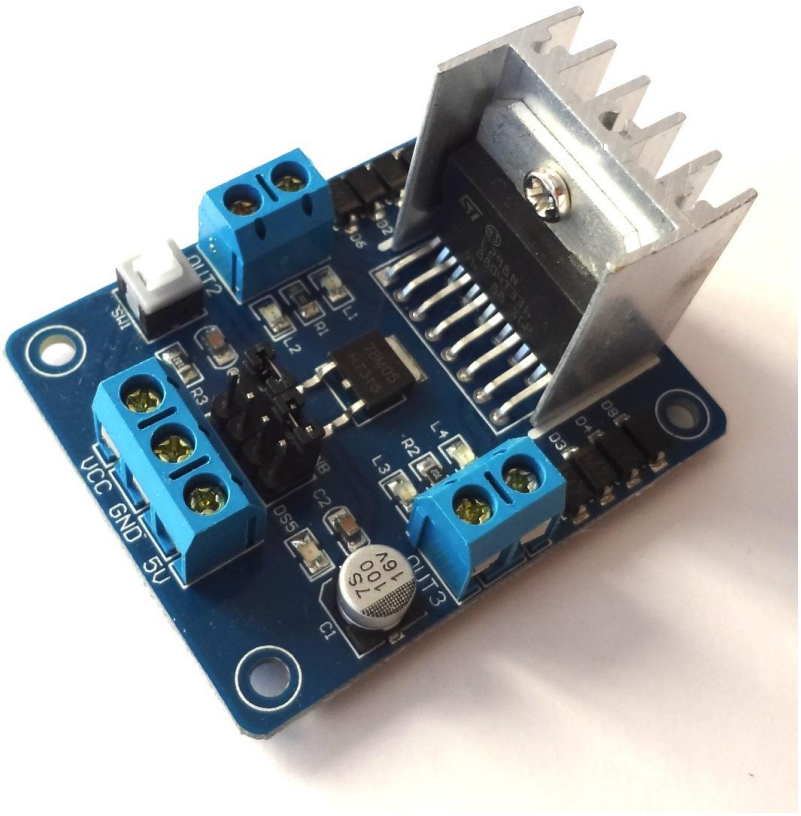


Figura 13 – Placa Ponte H
Fonte: Autoria própria.

Para o controle dos motores, foram utilizadas duas pontes H, baseadas no circuito integrado L298 [Figura 13], pois a corrente fornecida pelo Arduino não é suficiente para a operação do motor, sendo necessário o uso de uma fonte externa conectada diretamente à ponte. Para controlar cada motor de passo, são necessários quatro pinos digitais do Arduino conectados à placa da ponte H, além de um pino GND e outro pino fornecendo 5V para alimentar o circuito lógico da ponte. A sequência lógica necessária para o controle de passo inteiro do motor é representada no Quadro 1. A rotina de sequenciamento das bobinas pode ser feita pelo programador ou aproveitando-se de bibliotecas prontas, como a *Stepper.h*, fornecida gratuitamente junto à distribuição IDE do Arduino (MARGOLIS, 2011).

Passo	Fio 1	Fio 2	Fio 3	Fio 4
1	1	0	1	0
2	0	1	1	0
3	0	1	0	1
4	1	0	0	1

Quadro 1 – Sequência de controle dos motores de passo: 1 e 0 indicam se a linha está energizada ou desernegizada, respectivamente.
Fonte: Autoria própria.

3.2.5 Alimentação

Para a alimentação dos motores, foram conectadas à placa Arduino duas fontes externas. Uma é capaz de fornecer uma tensão de 6V e 2A de corrente para alimentar os Servos do projeto, e outra com tensão de 12V e 10A para alimentar os motores de passo, uma vez que a placa por si só não é suficiente para sustentar os motores.

A fonte externa de 6V foi ligada ao *Power Jack* da placa do micromanipulador, vinculada diretamente ao pino VIN, pino este conectado aos servos. Os pinos de *GND* dos servos e motores de passo foram ligados ao *GND* da placa. A fonte de 12V foi conectada diretamente aos circuitos das pontes H, responsáveis pela interface entre o Arduino e os motores de passo.

Para alimentação dos sensores, o pino 5V da placa foi suficiente. A utilização deste pino facilitou a leitura dos sensores de força, uma vez que a leitura feita pelas entradas analógicas compara a tensão de entrada com a tensão de referência da

placa. Adaptações seriam necessárias caso a tensão de alimentação dos sensores de força não fosse equivalente à da referência.

3.2.6 Garras

Para este projeto, foram desenvolvidas garras, cujas pinças são controladas por servos independentes, conectadas a uma base rotacionada por um motor de passo:

Foram criados diversos modelos e estudos, buscando criar uma base que atendesse aos requisitos do projeto. Apesar de existirem diversos modelos de garra disponíveis no mercado, muitos destes são movidos apenas por um único servo, que controla as duas pinças da garra simultaneamente [Figura 14]. Para dar um retorno de força específico para cada pinça, foram necessárias pinças cujas movimentações fossem independentes uma da outra.

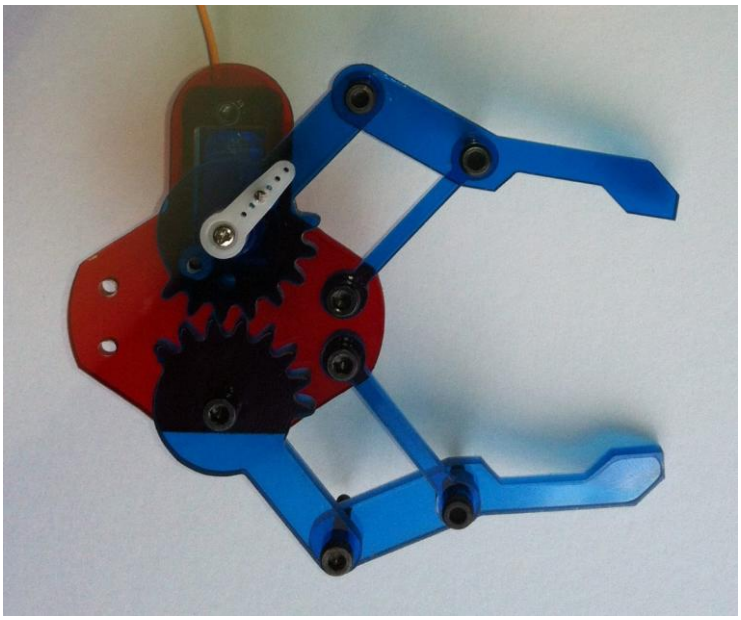


Figura 14 – Garra robótica ROB-10332
Fonte: Sparkfun, 2012.

A base, sobre a qual os servos foram acoplados, também deve ser capaz de rotacionar e captar as forças exercidas neste eixo de movimentação. Por isso foi criada com um encaixe para o sensor de força *Micro Load Cell*, empregando rolamentos (AST, 2014) de modo que a força exercida pelo motor de passo fosse

transmitida ao conjunto superior exclusivamente pelo corpo do sensor [Figura 15]. Deste modo, qualquer resistência à rotação ou força aplicada contra a base será captada pelo *MLC*.

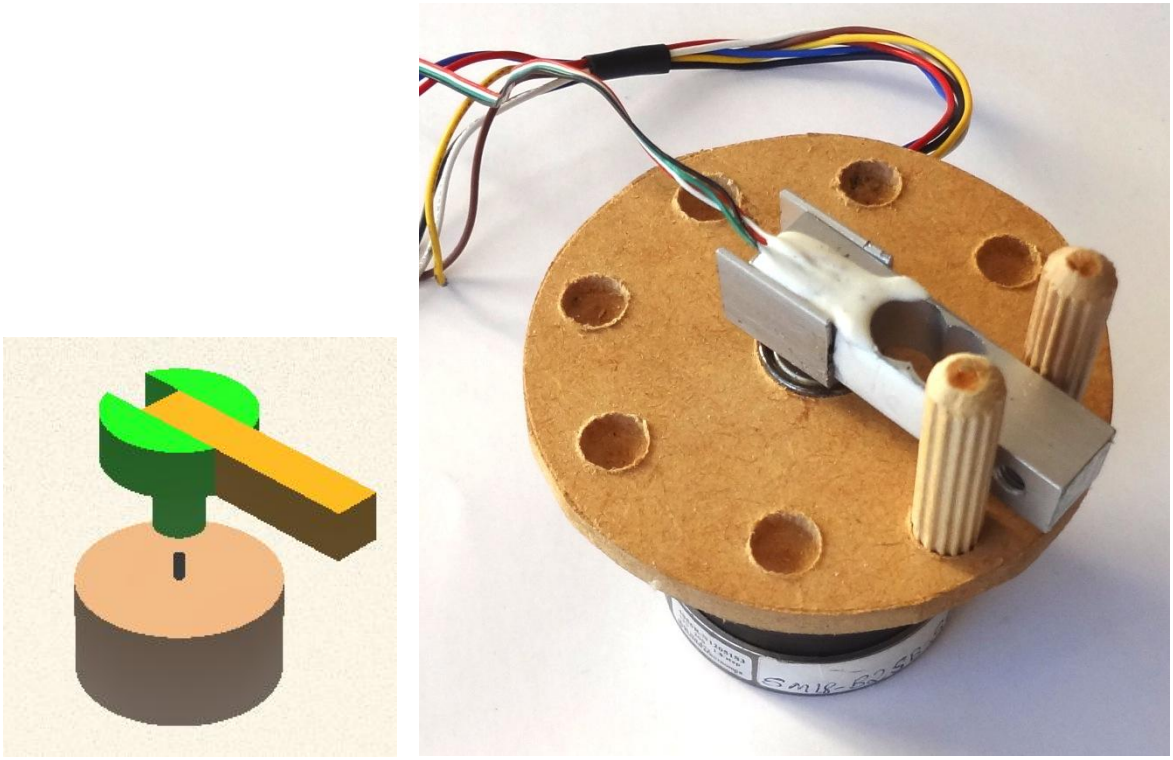


Figura 15 – Rolamento e sensor *MLC*: modelo e protótipo
Fonte: Autoria própria.

Os componentes foram todos modelados computacionalmente, utilizando-se o POV-RAY(POV RAY, 2014), um aplicativo em *software* livre, sendo o resultado apresentado na Figura 16. Em seguida, as medidas foram impressas em folhas sulfite para servir de gabarito no processo de usinagem. Grande parte das peças e componentes do protótipo foi confeccionada a partir de madeira MDF, conforme mostrado na Figura 17, por ser uma opção de baixo custo, leve, de fácil acesso e manuseio. Isso não descarta a possibilidade de utilização de outros materiais, escolha que pode variar, dependendo do foco de aplicação do projeto.

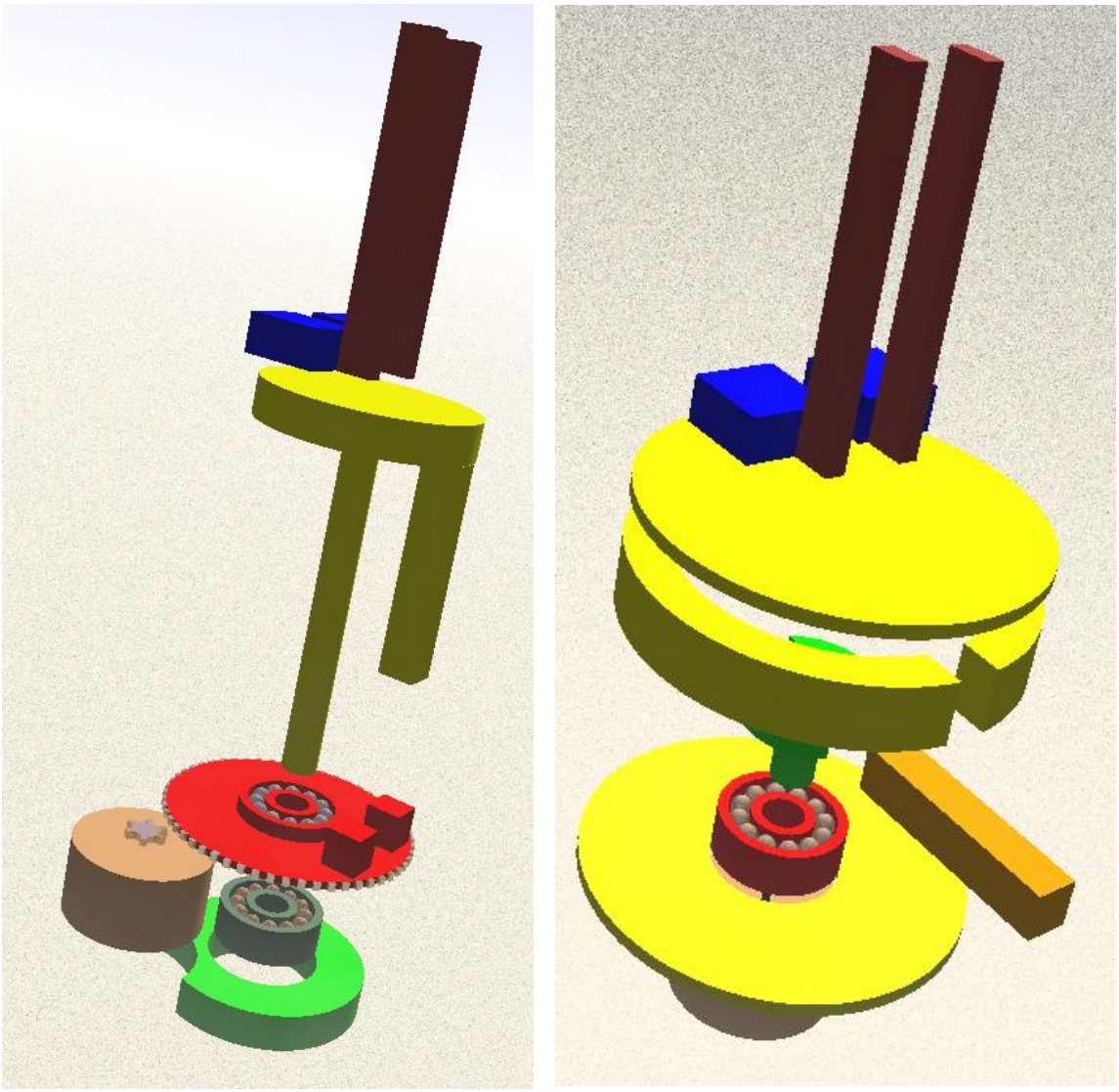


Figura 16 – Geração de modelos de garra: uma das versões antigas e modelo do protótipo final

Fonte: Autoria própria.

Foram produzidas duas garras, uma garra é controlada remotamente e outra é a adaptada para ser acoplada à mão do usuário. Para simplificar a etapa mecânica, neste projeto os modelos de garra utilizados para o manipulador e para a garra robótica foram os mesmos, mas um desenho não depende diretamente do outro, e as proporções podem ser alteradas.

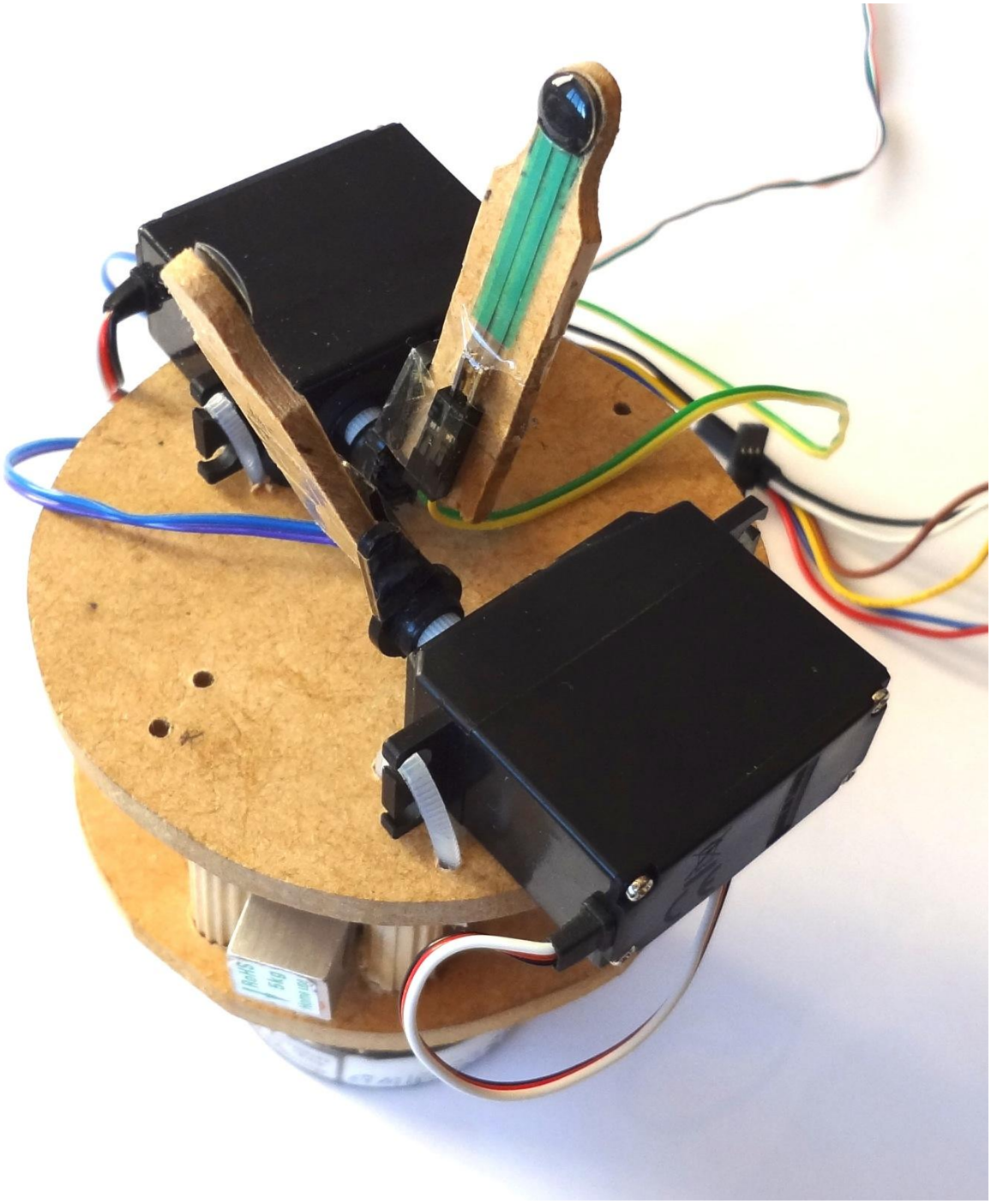


Figura 17 – Detalhe do protótipo da garra teleoperada
Fonte: Autoria própria.

3.3 MANIPULADOR

O manipulador consiste no conjunto de dispositivos, mecânicos e eletrônicos que interagem com a mão do operador, acompanhando a movimentação do seu polegar e indicador. Também tem o objetivo de fornecer a sensação de tato por meio

do retorno de força, de acordo com a leitura dos sensores presentes na garra robótica remota.

3.3.1 Sensores de posição e movimento

Neste projeto, foram utilizados dois servos motores munidos de garras e com um sensor de força cada, como pode ser observado na Figura 18.

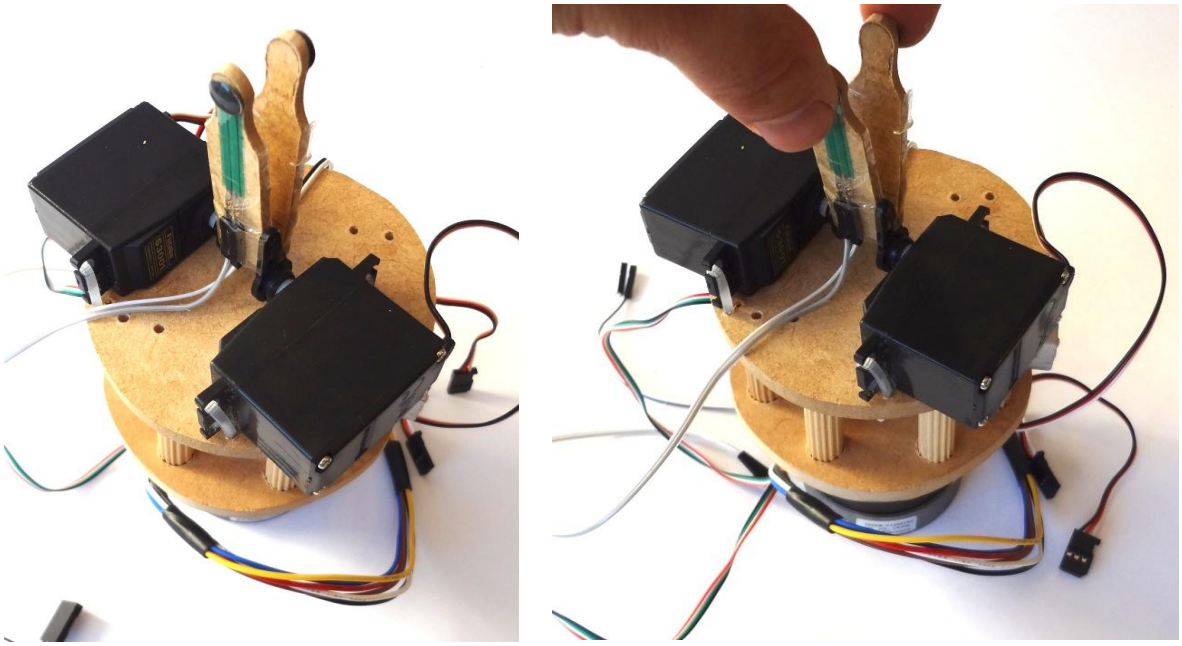


Figura 18 – Protótipo do manipulador: diferentemente da garra controlada remotamente, os sensores *FSR* do manipulador são posicionados na área externa das pinças, de modo a captar a força exercida pelo fechamento dos dedos do usuário
Fonte: Autoria própria.

O objetivo aqui foi captar a posição do dedo indicador, polegar e rotação de punho do usuário, replicando o movimento na garra robótica do manipulador. A movimentação de cada pinça da garra controlada remotamente está proporcionalmente relacionada ao posicionamento e movimentação dos servos presentes no manipulador.

Para captar a posição dos dedos do usuário, as pinças do manipulador buscam sempre manter contato com a mão do mesmo, de forma que se forme um tipo de conexão entre ambos. Isso é possível fazendo com que o manipulador sempre exerça uma pressão contrária à dos dedos do usuário. Se os dedos se movimentam, a leitura dos sensores de força é alterada, e os servos são acionados

para atualizar sua posição. Se o operador alivia a pressão contra o sensor, o servo recebe um comando para abrir até que a leitura mínima do *FSR* seja restabelecida, ou, se for feita uma força contrária, o servo fechará o manipulador.

Na lógica de controle, os valores de saída dos sensores *FSR* são comparados com dois limites, conforme ilustrado na Figura 2: um superior, referente à invasão e força mínima para se fechar o servo; e um inferior, referente à ausência e força máxima para se abrir o motor. Se o valor medido permanece entre ambos os limites, a garra permanece em repouso.

O movimento de rotação e posição do punho do usuário é medido pela leitura do *Micro Load Cell*, que fornece como saída uma tensão monitorada pelo micromanipulador Arduino. Quando não existe força sobre o mecanismo, o sistema permanece em repouso. Se uma força atua no sentido horário ou anti-horário, a tensão varia, de acordo com a direção e intensidade da força. Para acompanhar esta movimentação, o motor de passo é acionado.

Diferentemente do controle dos servos e sensores *FSR,* aqui não é necessário realizar buscas por contato, pois o acompanhamento da movimentação é feito de acordo com a força medida pelo *MLC* (*Referência*) comparado aos limites *GirarEsquerda* e *GirarDireita* [Figura 19]. Estes limites são obtidos préviamente à execução do sistema: são realizados testes nos quais não existem forças externas atuando no sistema, e, então, armazenam-se os valores dos sensores. Estas leituras são utilizadas nos cálculos da movimentação dos motores de passo, sendo comparadas com os valores obtidos durante a execução do protótipo: se o operador rotaciona a base, o valor do *MLC* do manipulador muda, podendo ultrapassar o limite superior ou inferior, de acordo com o sentido da movimentação.

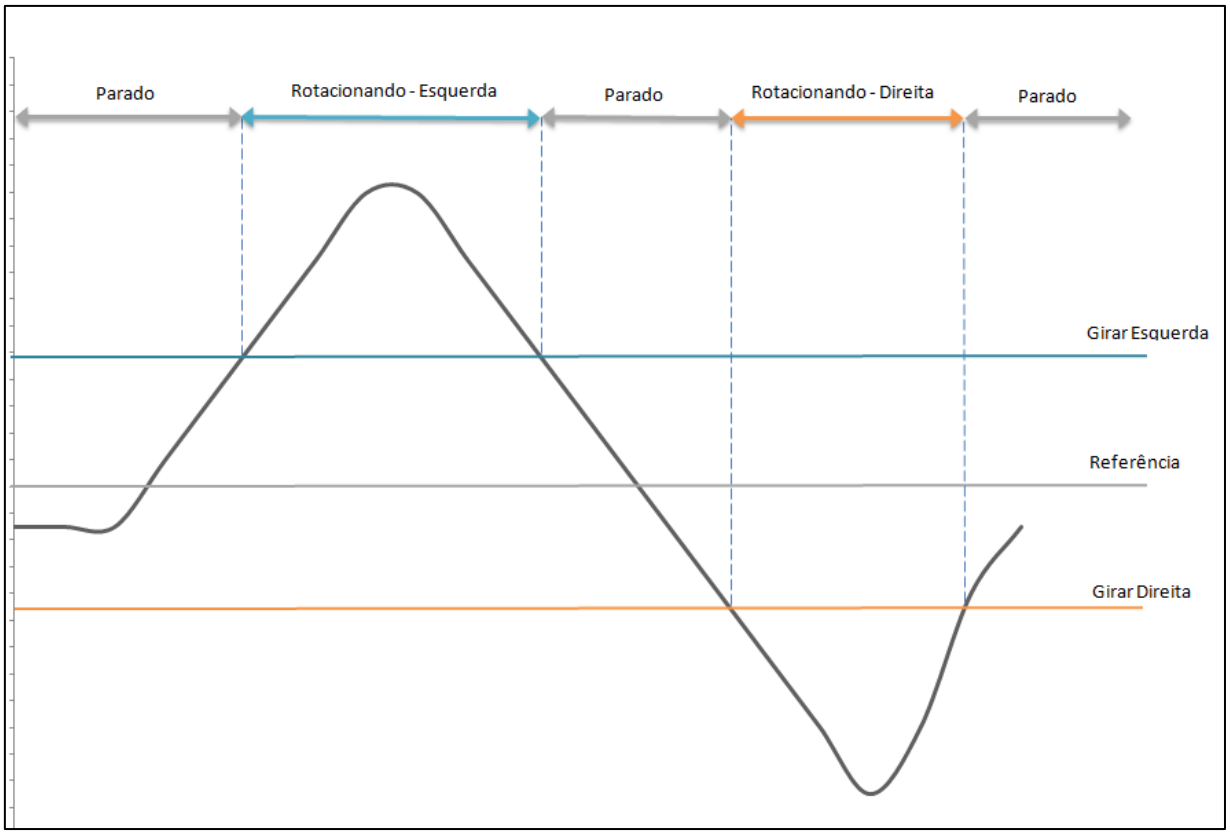


Figura 19 – Limites sensor *MLC* e movimentação dos motores de passo: Se o valor do sensor *MLC* (*Referência*) ultrapassar *GirarEsquerda* ou *GirarDireita*, os motores de passo movimentam-se. Do contrário, permanecem em repouso
Fonte: Autoria própria.

Muitas das variáveis utilizadas nos cálculos da movimentação do manipulador foram criadas com o objetivo de melhorar a interação entre usuário/ máquina, mas podem ser modificadas e melhoradas de acordo com a aplicação. Como exemplos, podem ser citadas as verificações de colisão baseadas no posicionamento das pinças, que previnem que uma tente “atravessar” a outra e ocasione danos ao protótipo; o número de passos dados pelo motor de passo é monitorado, para que não ocorra rotação ilimitada do motor; um intervalo mínimo de tempo foi adicionado para a tentativa de inversão da movimentação dos motores, buscando reduzir o *jitter* causado pela perda de contato entre mão e manipulador durante a movimentação, dentre outros.

3.4 GARRA ROBÓTICA

3.4.1 Mecanismos de movimentação e posicionamento

Com o objetivo de mover a garra de acordo com a movimentação da mão do usuário surgem algumas possibilidades. Uma delas é a variação angular dos servos da garra. Ela pode ser igual, maior ou menor do que as do manipulador, de acordo com a sensibilidade e aplicação desejadas. Isso é possível por meio da análise dos limites angulares medidos pelo manipulador e os limites desejados para a garra robótica, o que possibilita remapear uma faixa de valores para outra pelo mesmo cálculo utilizado na etapa de calibragem:

$$f = (x - in_{min}) \times \frac{(out_{max} - out_{min})}{(in_{max} - in_{min})} + out_{min}$$

Sendo: x a variável de entrada a ser mapeada no novo intervalo, in_{min} e in_{max} os limites da faixa de valores de entrada do manipulador, e out_{min} e out_{max} os limites desejados para a faixa de valores de saída para a garra robótica.

Os sensores de força presentes nas garras telecontroladas não influenciam diretamente na movimentação da mesma, sendo utilizados nos cálculos de movimentação dos servos do manipulador que acompanham a movimentação do usuário, e só então, de acordo com o posicionamento após os cálculos de remapeamento de valores, as posições dos servos da garra robótica são atualizadas.

3.5 REALIMENTAÇÃO DE FORÇA

Com a finalidade de realizar o retorno de força captada pela garra teleoperada ao usuário, foram analisadas as saídas dos sensores de força e criada uma resposta por meio da movimentação dos motores. Para simplificar o problema, este foi dividido em duas frentes: resistência e força contrária.

Se for detectada uma variação de pressão pelos sensores da garra robótica, esta sensação deve ser passada ao usuário do manipulador. A ideia aqui foi criar uma resistência à ação de fechar ou rotacionar a mão, proporcional à resistência

encontrada pela garra robótica. Quando o operador fecha o manipulador, uma verificação é realizada, e o valor da leitura do sensor *FSR* do manipulador é comparado ao da garra robótica. Se a pressão exercida sobre os sensores não for superior ao da garra robótica multiplicada por uma constante de proporção (menor, maior ou igual à força do usuário), a garra não irá se fechar.

Quando a garra robótica sofre ação de uma força externa, deve-se gerar uma força proporcional na mão do operador para que ele perceba a mesma. Isso é realizado por meio do monitoramento dos sensores da garra robótica: se a força lida pela garra robótica é maior que a lida pelo manipulador, os valores de pressão mínimos utilizados pela busca de contato pelos servos são atualizados, assim como os valores médios do motor de passo, movimentando o manipulador para que a força seja percebida pelo operador.

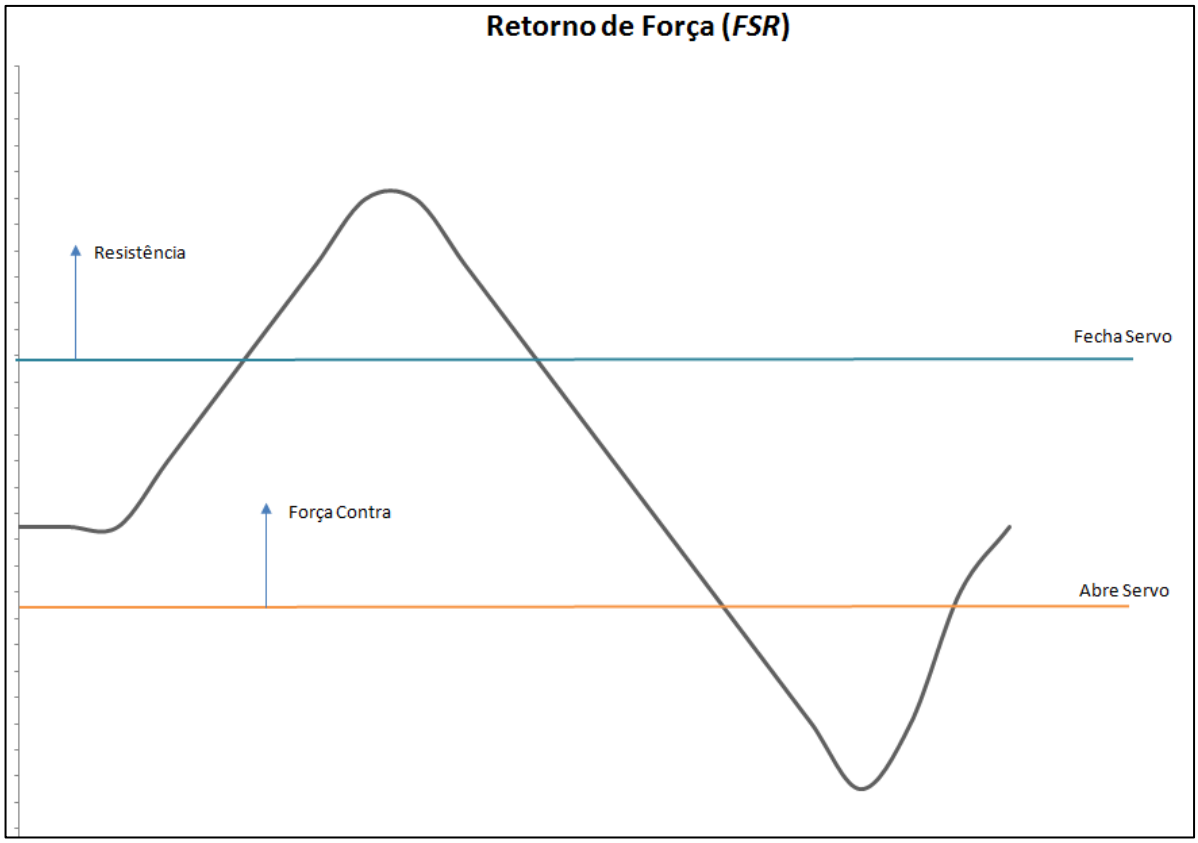
Nas Figura 20a e 20b, estão representadas as atualizações dos limites utilizados na interpretação das leituras dos sensores presentes no manipulador, de acordo com a leitura dos sensores instalados na garra operada à distância.

No manipulador, quando é detectada uma força no sensor *FSR* da garra teleoperada, os limites de invasão e ausência utilizados nos cálculos da movimentação são incrementados proporcionalmente. Para fechar a pinça, o operador deverá aplicar uma força maior que o novo limite, e, se estiver em repouso, perceberá uma força contrária, devido à alteração do valor mínimo de ausência, que tem como consequência a abertura da pinça.

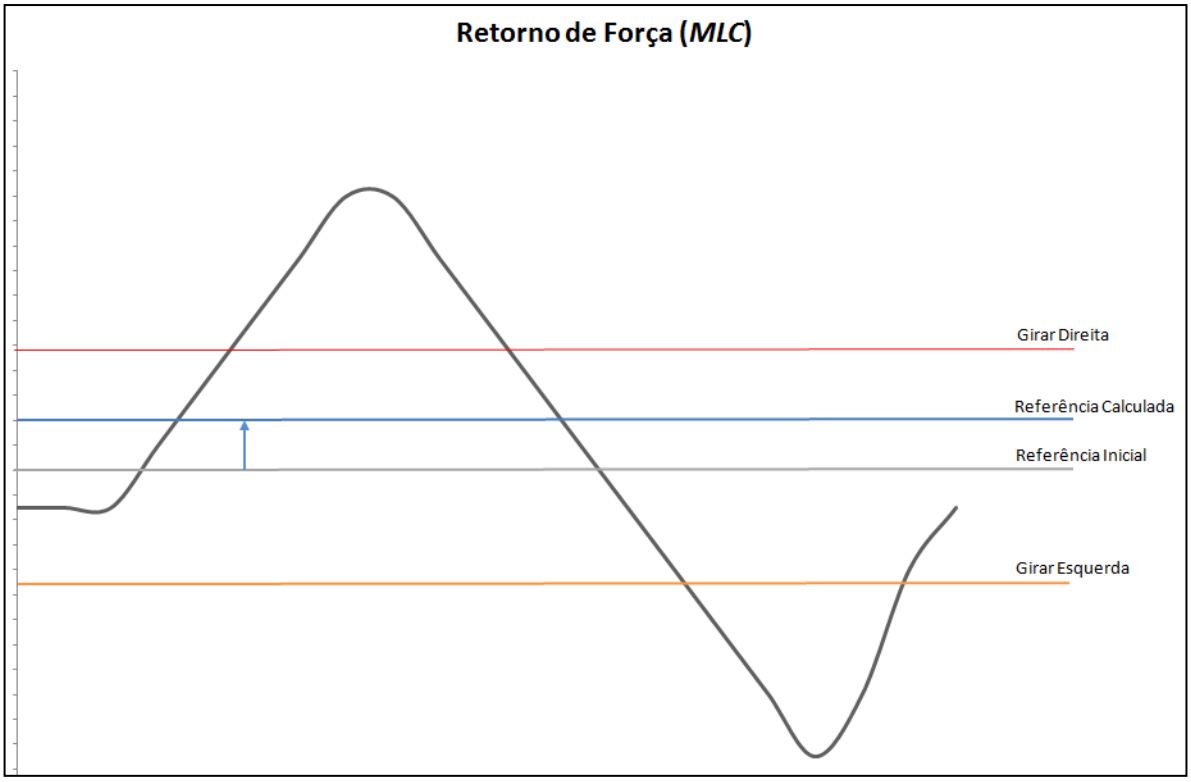
O programa realiza o cálculo do retorno de força referentes à rotação da base e as leituras dos *MLC*s. Se existir alguma resistência mecânica, um novo valor de referência, $RefCalculada$, é obtido durante os cálculos da movimentação.

$$RefCalculada = Diferença(LeituraMlcLuva, RefMlcLuva) + Diferença(LeituraMlcGarra, RefMlcGarra)$$

Sendo: $Diferença$ uma função que retorne $LeituraMlc - RefMlc$, caso a leitura do *MLC* seja maior ou igual à da referência, e $RefMlc - LeituraMlc$, caso contrário. Se $RefCalculada$ for maior que o limite *GirarDireita* ou *GirarEsquerda,* presentes na Figura 20b, os motores de passo irão se movimentar.



(a)



(b)

Figura 20 – Retorno de força – Se forças são detectadas pelo sensor remoto: a) *FSR*: os limites *FechaServo* e *AbreServo* são incrementados; b) *MLC*: uma nova Referência é calculada, levando em consideração a leitura da garra teleoperada
Fonte: Autoria própria.

3.6 FIRMWARE

A programação de *firmware* foi feita no ambiente de desenvolvimento Arduino, cuja linguagem de programação baseia-se no Wiring (WIRING, 2014), e pode ser dividida em três partes principais: estruturas, variáveis e funções.

A Figura 21 apresenta o diagrama de fluxo do *firmware*. Quando o programa é inicializado, a placa é preparada para controlar os motores e realizar a leitura dos sensores. Feito isso, o sistema entra em *loop*, obtendo o valor dos sensores, realizando os cálculos necessários e atualizando as posições dos motores de acordo com a lógica elaborada. Se nenhuma resistência mecânica for detectada, os motores são acionados para que o protótipo acompanhe a movimentação do usuário ou, caso exista força externa atuando na garra remota, são controlados de modo a oferecer realimentação de força ao usuário.

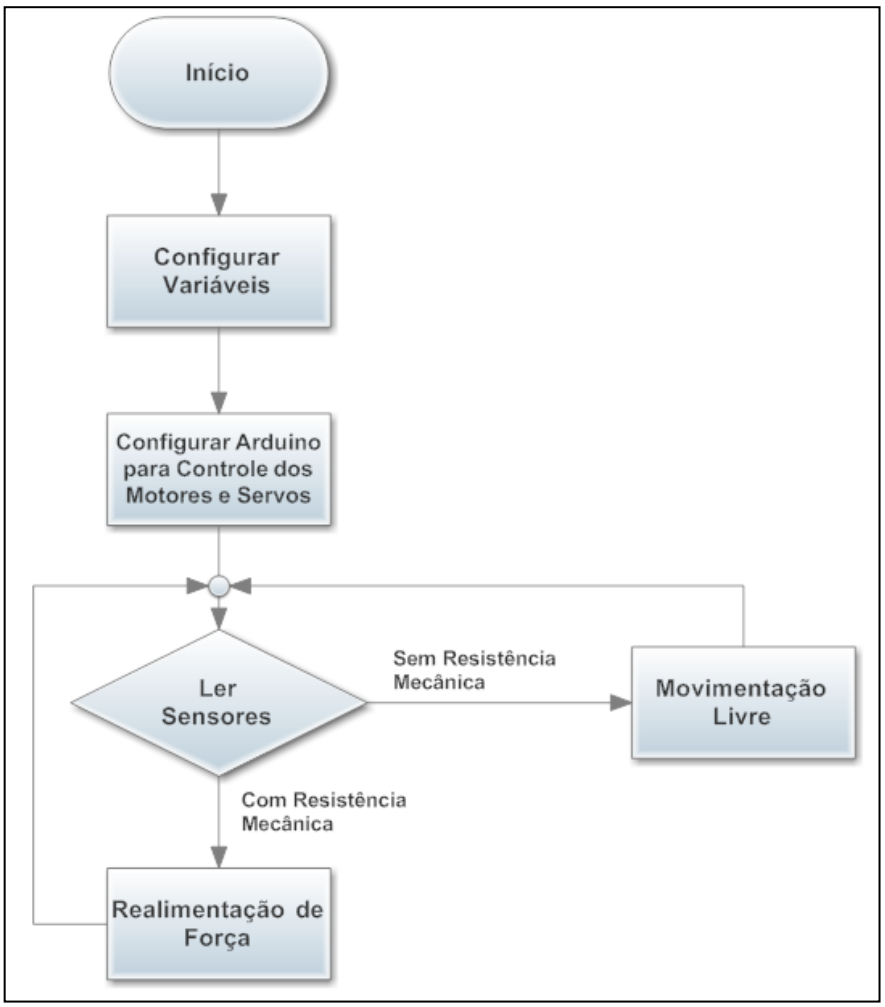


Figura 21 – Diagrama de fluxo
Fonte: Autoria própria.

Alguns dos valores definidos inicialmente são: forças relativas à movimentação do manipulador; posições iniciais dos servos; e tempo de atualização dos motores e sensores, dentre outros. No laço que se repete, são verificadas a ocorrência de eventos, as variáveis de posicionamento dos motores e as informações de retorno de força são atualizadas, de acordo com as leituras dos sensores, tendo três estados possíveis:

- **Movimentação livre**: não é lida nenhuma resistência mecânica por parte dos sensores da garra, o manipulador acompanha a movimentação do operador, e a posição da garra é atualizada.
- **Resistir fechamento**: Uma resistência mecânica foi encontrada pela garra na ação do usuário fechar a pinça ou rotacionar punho: o valor mínimo de força necessário para o operador fechar ou rotacionar o manipulador é atualizado, baseando-se nas leituras da garra remota.
- **Força externa**: Uma força externa é aplicada à garra, independentemente da movimentação desta. Para os servos e sensores *FSR*, atualiza-se a variável auxiliar de busca de contato do manipulador com um novo valor mínimo proporcional ao lido pela garra. Para os motores de passo e MLCs, altera-se o valor médio utilizado nos cálculos da interpretação da leitura e rotação do motor.

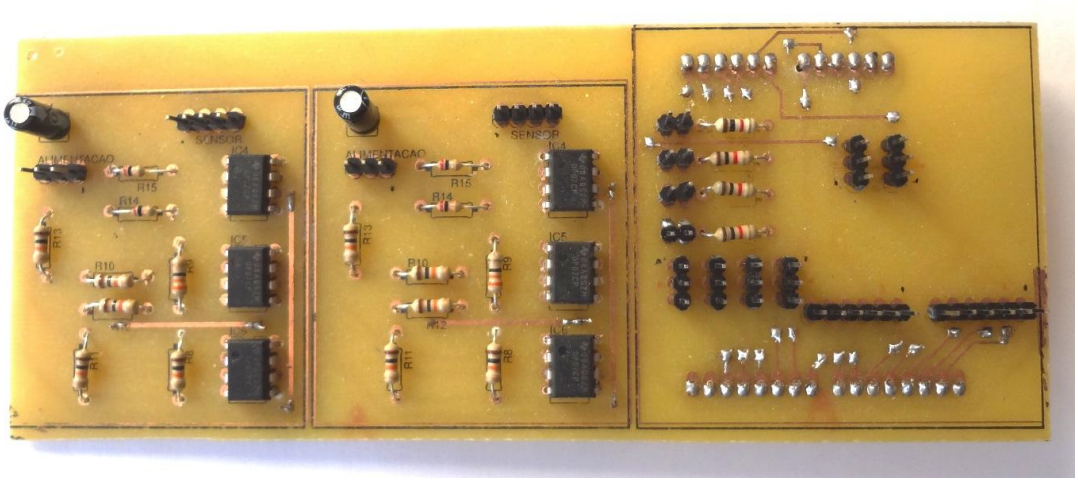
O primeiro item é responsável pela detecção da movimentação do usuário e replicação na garra, enquanto que o segundo e terceiro são relativos à realimentação de força.

3.7 SISTEMA COMPLETO

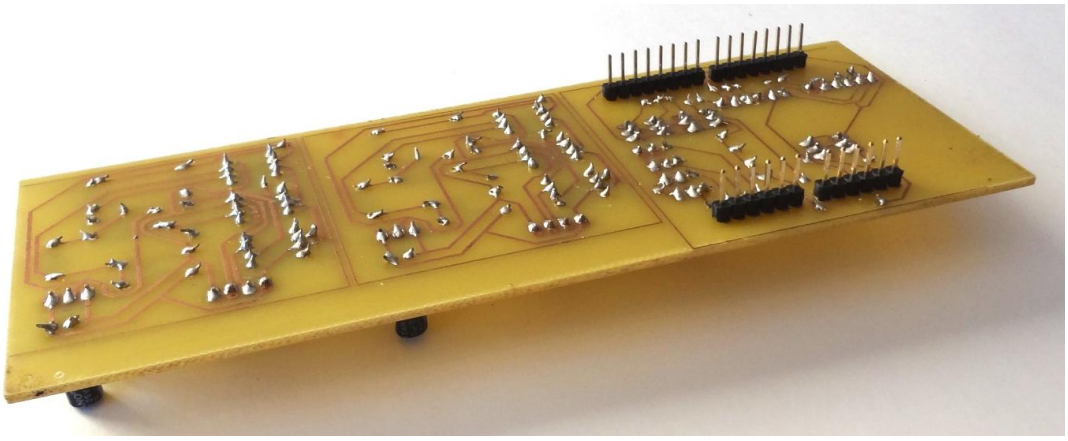
O micromanipulador Arduino é responsável pelo controle e parte lógica do sistema, devendo ser capaz de realizar a leitura dos sensores, processar os dados e enviar os sinais necessários à movimentação dos motores. Com o objetivo de organizar, reduzir o conjunto de cabos e conexões do protótipo e minimizar problemas relacionados ao mau contato entre conectores, dentre outras vantagens, foi desenvolvida uma placa baseada no Arduino *Sensor Shield* V4.

A placa, mostrada na Figura 22, pode ser subdividida em três circuitos independentes: o primeiro, conectado diretamente à placa, contém os pinos de

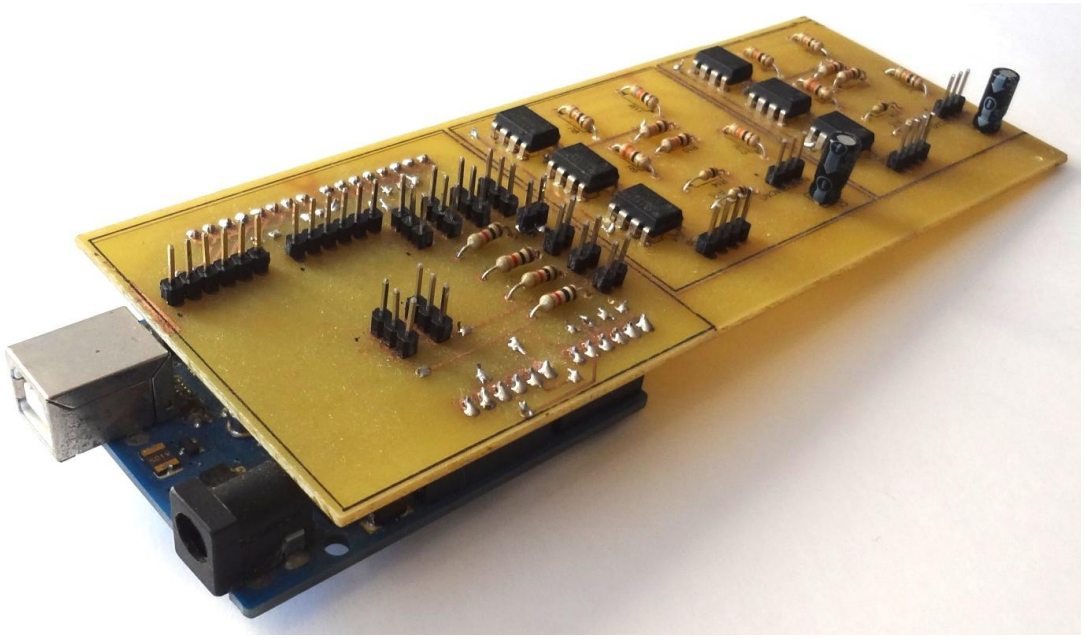
encaixe para os sensores e comunicação com os motores. Os dois últimos circuitos são utilizados na leitura dos *MLCs,* contendo os circuitos do amplificador operacional *OP07CP*. As trilhas foram desenhadas com o auxílio do *software* Eagle 6.5.0 (MONK, 2014) e posteriormente impressas por uma impressora a laser para serem utilizadas na confecção das placas por transferência térmica. Os desenhos das placas desenvolvidas estão disponíveis no Anexo B.



(a) Vista Frontal



(b) Verso



(c) Placa acoplada ao Arduíno

Figura 22 – Placa desenvolvida com o amplificador e a interface para os sensores, servo motores e motores de passo
Fonte: Autoria própria.

Em relação ao funcionamento do manipulador, este módulo detecta as forças e calcula as posições do dedo indicador e polegar do usuário. O protótipo identifica também a rotação do punho por meio de um sensor de força *MLC*, fornecendo uma resposta ao usuário conforme o estado da garra operada remotamente. A garra robótica possui uma mecânica semelhante à do manipulador, diferindo da primeira na localização dos sensores *FSR*, posicionados na parte interna da garra e não externa, como a do operador. Quando o conjunto é acionado, uma rotina de inicialização é executada, e a placa se prepara para a leitura dos sensores e a movimentação dos motores. Na sequência, os servos são colocados em uma posição inicial pré-definida, e, terminado o processo de inicialização, os sensores são lidos, e o comportamento do conjunto é atualizado.

O funcionamento padrão do manipulador é o de acompanhar o movimento de pinça dos dedos indicador e polegar, assim como acompanhar a rotação da mão do usuário. Por sua vez, a garra teleoperada segue a movimentação proporcionalmente. Se a garra remota encontra resistência mecânica, os parâmetros de movimentação do manipulador são alterados, para que o usuário tenha retorno da resistência mecânica à movimentação ou força contra a garra remota.

A Figura 23 apresenta a última versão do protótipo, com todos os componentes conectados:

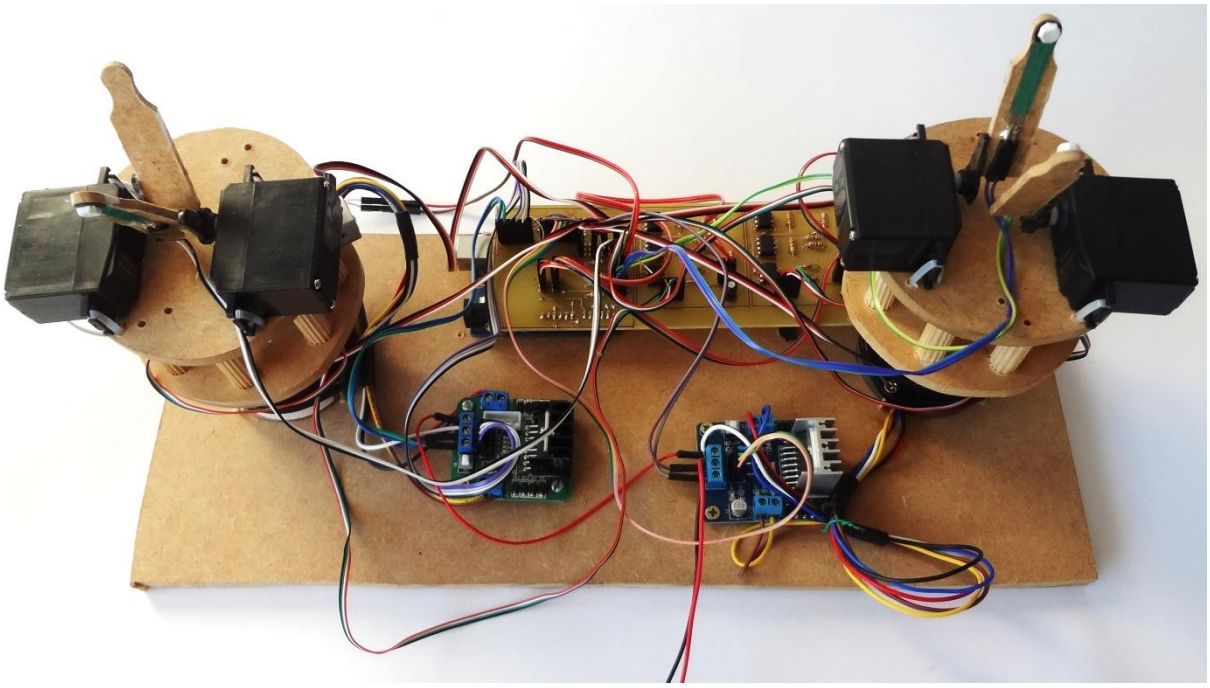


Figura 23 – Protótipo final: Manipulador à esquerda e Garra Remota à direita
Fonte: Autoria própria.

4 ANÁLISE DOS RESULTADOS

O protótipo obteve os resultados e comportamentos esperados, apresentando algumas limitações, entre elas as de cunho mecânico e as causadas pela faixa limite de funcionamento dos motores e sensores utilizados, tópicos abordados neste capítulo após a descrição e análise dos testes realizados.

Os primeiros servos utilizados (TP9G SG90 e TP MG995) não apresentaram uma movimentação suave e precisa, oferecendo um baixo limite de força e torque, além da faixa de operação e posicionamento em resposta aos pulsos PWM de entrada variar para cada modelo de servo. De qualquer modo, o uso inicial destes modelos de baixo custo foi suficiente para verificar os conceitos aplicados.

O teste individual do servo TP9G SG90 envolveu a análise da movimentação obtida em resposta às diferentes entradas PWM. Também foi possível alterar o servo para leitura do seu potenciômetro interno, o que acabou não sendo aproveitado pelo projeto, optando-se pelo ajuste manual prévio dos servos (capítulo 3.2.3).

O servo TP MG995, outra opção de baixo custo testada, não apresentou desempenho consideravelmente superior em relação ao microservo anterior. Apesar de apresentar maior capacidade de torque, este servo peca em precisão, exibindo problemas de *overshoot* com frequência: muitas vezes, ao se dirigir a uma nova posição, o servo a ultrapassa e tem que retornar. Outra inconveniência apresentada pelo modelo comparado aos outros testados foi a inversão da direção tomada pelo servo em relação ao PWM de entrada, dificuldade que pode ser contornada via *software*.

Os servos utilizados pelo protótipo final foram os do modelo Futaba S3001, cujo desempenho foi superior aos servos testados anteriormente, seja em relação à velocidade, torque ou precisão. Uma possível desvantagem deles, dependendo da aplicação e quando comparado ao micro servo, é o seu maior tamanho, que exigiu que as bases da garra robótica e do manipulador fossem construídas com um diâmetro também superior, para que dois servos pudessem ser suportados por cada uma delas.

Apesar de também ser possível a utilização de servos na movimentação da base do manipulador e da garra robótica, foram utilizados motores de passo. Esta

escolha foi feita visando expandir as possibilidades do projeto e viabilizar o estudo e análise de uma nova opção de motor.

Os componentes do manipulador têm grande influência sobre a qualidade de captação da movimentação e retorno de força ao usuário. Quanto maior a distância entre o ponto de contato do usuário com o manipulador, em relação ao eixo de movimentação, e quanto maior o ângulo rotacionado pelo motor, menos suave será a movimentação para o usuário. Isso pode ser representado com o cálculo do comprimento de um trecho de circunferência:

$$L = \frac{\theta \times \pi \times R}{180}$$

Sendo: θ o ângulo em graus rotacionados por passo pelo motor e R o raio. Quanto maior o ângulo e o raio, maior o arco L de saída. Para reduzir o arco L, é necessário buscar motores com um menor ângulo rotacionado por passo, ou reduzir a distância entre a extremidade e o eixo de rotação.

Durante a operação do sistema, as leituras dos sensores são constantemente atualizadas. O circuito *FSR* gera como saída uma tensão inversamente proporcional à resistência do sensor*,* podendo ser captada pela entrada analógica da placa Arduino. Como a resposta do sensor é aproximadamente inversa à força aplicada, o resultado final é diretamente proporcional à tensão e força. Assim, o circuito apresenta uma saída quase linear em resposta à força aplicada sobre o sensor (INTERLINK ELETRONICS, 2010), como pode ser observado na Figura 6, facilitando a leitura e calibragem do conjunto.

Os sensores *FSR* foram utilizados para obtenção de dados relativos às pressões aplicadas contra as garras utilizadas pelo projeto. Seu funcionamento pode variar de acordo com o ambiente e superfície na qual se encontra acomodado, e, dependendo da distribuição da pressão sobre a sua superfície, os resultados podem ser distintos. Para otimizar o funcionamento do sensor, o conjunto foi testado em superfícies planas para evitar possíveis danos ao sensor. Existem outros formatos do sensor disponíveis no mercado, sendo que o modelo utilizado deve ser escolhido com base na superfície na qual será instalado e do objeto a ser manuseado. Uma das dificuldades encontradas na utilização deste sensor no projeto foi a de manter as forças de pinça aplicadas pela mão do usuário e garra robótica apenas nas áreas

sensíveis do sensor. Nos casos de contato na borda ou fora do sensor, as leituras dos posicionamentos, assim como retorno de força, são comprometidas.

Sensores do tipo *MLC* são comumente encontrados em balanças e se revelaram opções precisas e baratas, apesar de seu maior tamanho, quando comparados aos *FSR.* Para calibrá-los, a saída do sensor foi medida enquanto o sistema era mantido em repouso. Quando a base da garra é tensionada, este valor varia para mais ou para menos, de acordo com a direção da força. A partir da leitura do sensor acoplado ao manipulador, o motor de passo pode ser acionado para acompanhar a movimentação do usuário. Já a saída do *MLC* da garra telecontrolada é aproveitada para alimentar o retorno de força.

Algumas das dificuldades relacionadas à movimentação dos motores, com base na leitura dos sensores, foram: as perdas de leitura causadas pela perda de contato entre o sensor *FSR* e o operador; forças captadas pelo sensor *MLC* causadas pela movimentação da base e não diretamente relacionadas ao usuário, tais como vibrações. Buscando reduzir estes problemas, algumas soluções foram testadas, dentre elas:

- Aplicação de filtro de média à saída dos sensores;
- Criação de intervalos de tempo ou atrasos entre as leituras e atualizações do posicionamento dos motores.

A adição de um filtro de média não trouxe benefícios consideráveis ao desempenho do sistema e, dependendo dos parâmetros utilizados, chegou a causar um atraso significativo e não desejado entre a leitura do comando do operador e a movimentação da garra.

A calibragem baseada na variação intervalos de tempo apresentou melhores resultados: quanto mais elevada a taxa de atualização dos motores, mais ágeis se tornam as atualizações dos posicionamentos e movimentação do protótipo, com a desvantagem de aumentar o aparecimento de *jitter*, devido à maior probabilidade de ocorrência de ruídos mecânicos e detecção de leituras espúrias pelo sensor. A aplicação de taxas demasiadamente baixas resultam em atrasos às respostas do sistema em relação às entradas dos sensores. Equilibrando estes dois fatores, é possível fornecer uma resposta confortável ao usuário, minimizando os retornos indesejáveis e mantendo os intervalos de resposta em níveis não perceptíveis pelo operador.

Os módulos foram testados individualmente antes de serem conectados uns aos outros. Um dos primeiros testes envolveu a captura da movimentação do usuário pelo manipulador. Cada pinça da garra deve ser capaz de manter contato com os dedos do usuário, abrindo até que um valor mínimo na leitura da força sobre os sensores *FSR* seja atingido. A garra foi capaz de acompanhar a movimentação da mão durante os testes, excluindo os casos nos quais a velocidade da mão ultrapassou a velocidade máxima do servo.

Para futuros projetos, é possível estimar a velocidade com que o operador fecha ou abre seus dedos com base na variação da pressão medida, o intervalo entre medições e o comprimento da garra. Isso pode ser utilizado para aperfeiçoar a leitura dos movimentos do usuário.

Durante os testes, os componentes de madeira *MDF* utilizados pelo manipulador e pela garra telecontrolada não apresentaram problemas. O protótipo não foi mantido sobre condições climáticas agressivas, como umidade excessiva e exposição direta do sol, e não foi testada a capacidade máxima de carga suportada, com a aplicação de forças mais elevadas, por exemplo. Foram aplicados testes suficientes para validar o projeto, mas evitando danificar o mesmo.

Alguns ajustes na parte mecânica do projeto foram necessários para aperfeiçoar o funcionamento do protótipo. Os servos devem ser posicionados na base, de modo que o eixo da garra se encontre o mais próximo possível do eixo de rotação do motor de passo. Isso é necessário para que o sensor *MLC* seja capaz de captar as forças de torção de punho do usuário com maior precisão.

Para cada uma das garras foi adicionada uma superfície rígida, cujos principais pontos de contato com a pinça são: um junto à base da garra e outro em contato com o sensor, visando concentrar a força exercida pelo operador nos sensores *FSR* e não nas extremidades de madeira (Figura 24).

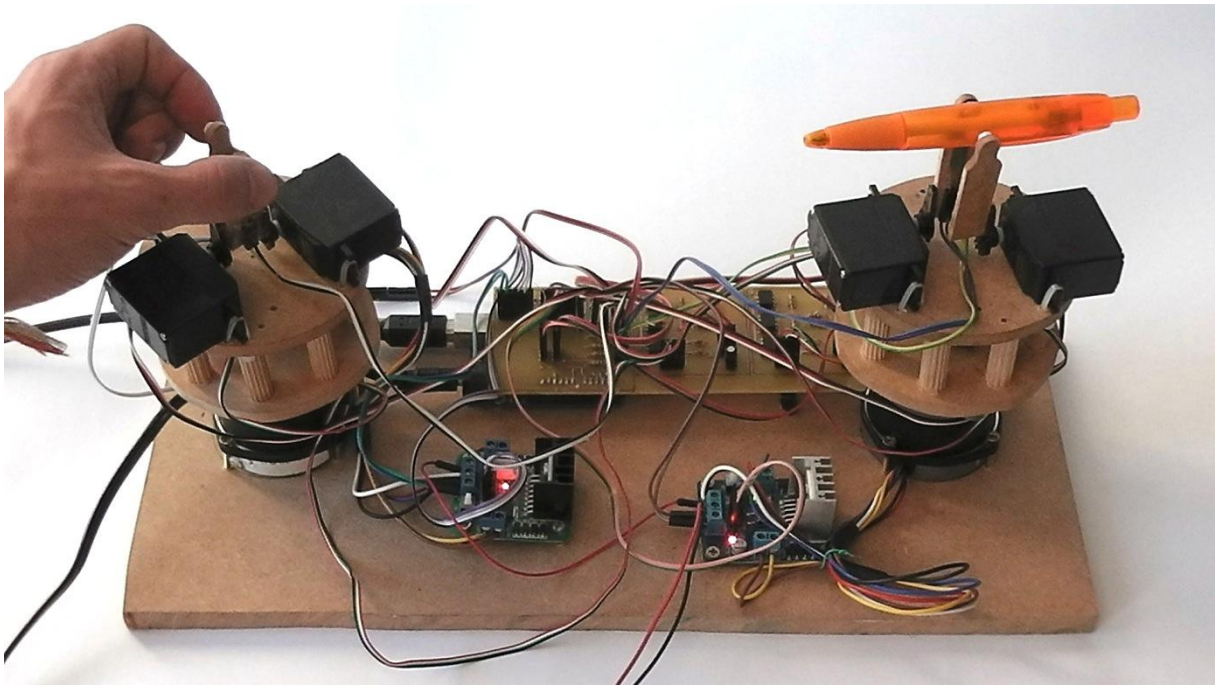


Figura 24 – Garra em ação
Fonte: Autoria própria.

O Arduino Uno foi suficiente para atender as demandas deste protótipo, mas também é possível utilizar a placa desenvolvida em outros modelos, se estes possuírem dimensões e propriedades semelhantes, como a placa do micromanipulador Arduino Mega, por exemplo.

A fim de aperfeiçoar a execução do código pelo micromanipulador, não foram empregadas funções de atraso como o *Delay* disponível na biblioteca Arduino. Foi utilizada a biblioteca aberta *TimeAction* (BREVIG, 2011) que simula o funcionamento de *Threads* (DEITEL, 2005). Deste modo, foi possível definir em milissegundos o intervalo de chamada de cada função, como controle dos motores ou sensores, sem ocupar o processador entre as chamadas.

5 CONSIDERAÇÕES FINAIS

No presente projeto, foi desenvolvido um protótipo funcional de uma garra robótica teleoperada por uma mão humana e com realimentação de força, atendendo ao objetivo geral, assim como os objetivos específicos de construir o manipulador e o mecanismo da garra robótica remota, implementar sensores de força, posição e *firmware* de controle. O protótipo funcionou corretamente e mostrou-se com grande potencial, seja no estudo de programação, de robótica ou para a implementação em situações reais com os devidos aprimoramentos.

O manipulador teve suas dimensões baseadas na mão do projetista e em função dos componentes utilizados. A mecânica pode ser aperfeiçoada para aceitar ajustes, a fim de se adaptar a diferentes operadores e funções. Apesar do protótipo utilizar o mesmo desenho no manipulador e na garra robótica, o tamanho e materiais desta última não dependem da dimensão do manipulador e podem ser substituídos por outros materiais e construídos em diferentes dimensões para melhor atender a aplicação na qual a garra será empregada.

Os componentes adquiridos e empregados neste projeto foram selecionados a partir da análise do custo/benefício proporcionado, entretanto, por valores maiores, podem ser encontrados modelos com desempenhos superiores aos utilizados. A escolha vai depender dos requisitos da aplicação do conjunto: a operação de objetos pequenos e/ou delicados requer motores de alta precisão, enquanto que a manipulação de objetos pesados requer modelos com maior capacidade de torque instalados na garra robótica, bem como uma reavaliação dos materiais e componentes mecânicos utilizados.

Os sensores também possuem uma faixa de força na qual trabalham (3.2.2), assim como diferentes formatos e área de contato. Podem ser encontrados modelos com desenhos distintos e outras faixas de leitura de pressão e precisão. Uma ideia para futuros projetos seria juntar a funcionalidade dos *strain gauges* utilizados pelo *MLC* na mesma base mecânica da garra, sem que haja a necessidade de empregar dois componentes independentes, procurando minimizar problemas com desencaixe do sensor, limitações causadas pela região de contato sensível do sensor, dentre outros, e também ampliar as possibilidades de captação de força.

Para o presente projeto foram desenvolvidos códigos tanto para testes individuais quanto para controle do conjunto manipulador/garra telecontrolada.

Melhorias podem ser feitas também em relação ao *firmware*, utilizando algoritmos de controle mais complexos e, se necessário, substituir o micromanipulador por outro modelo com maior capacidade de processamento.

Outros possíveis aprimoramentos e vertentes do projeto podem ser desenvolvidos no campo de alimentação ou transmissão de dados, arquitetando um sistema com comunicação sem fio e alimentado por baterias para trabalhar a longas distâncias, por exemplo.

REFERÊNCIAS

ACTION TECHNOLOGY 2010. **Datasheet: Step SM1.8B2SBSE**. Disponível em: <www.actiontechnology.com.br>. Acesso em: 12 fev. 2014.

ADEE, S. **Dean Kamen's "Luke Arm" Prosthesis Readies for Clinical Trials**. Disponível em: <spectrum.ieee.org/biomedical/bionics/dean-kamens-luke-arm-prosthesis-readies-for-clinical-trials>. Acesso em: 1 out. 2010.

AST 2014. **Datasheet: Ball Bearings - EZO R4AZ**. Disponível em: <www.astbearings.com>. Acesso em: 7 jun. 2014.

BREVIG, Alexander. **TimedAction Library for Arduino.** Disponível em: <playground.arduino.cc/Code/TimedAction>. Acesso em: 5 jul. 2014.

BUERGER, S.; HOGAN, N. Novel actuation methods for high force haptics. In: ZADEH, Mehrdad Hosseini. **Advances in Haptics**. Vukovar: Intech, 2010.

CHIU, Y. **Exoskeletons Are on the March**. Disponível em: <spectrum.ieee.org/robotics/medical-robots/exoskeletons-are-on-the-march>. Acesso em: 1 out. 2010.

DEITEL, Harvey; DEITEL, Paul. Multithreading. In: DEITEL, Harvey; DEITEL, Paul. **Java Como Programar**. 6. ed. São Paulo: Pearson Education, 2005. Cap. 23, p. 787.

FISHEL, Jeremy A.; LOEB, Gerald E. **Bayesian exploration for intelligent identification of textures**. Frontiers In Neurorobotics, Los Angeles, v. 6, n. 4, p.1-20, jun. 2012.

INTERLINK ELETRONICS 2010. **Datasheet: FSR Force Sensing Resistors Integration Guide**. Disponível em: <www.interlinkeletronics.com>. Acesso em: 23 jan. 2014.

MARGOLIS, Michael. **Arduino Cookbook**. 2. ed. Sebastopol: O'Reilly, 2011.

MONK, Simon. **Make Your Own PSBs with EAGLE**. 1. ed. Columbus: McGraw-Hill Education, 2014.

NATIONAL INSTRUMENTS. **Medindo distensão com Strain Gauges**. Disponível em: <www.ni.com>. Acesso em: 7 jun. 2014.

PATOGLU, V.; SATICI, A. Optimal design of haptics interfaces. In: ZADEH, Mehrdad Hosseini. **Advances in Haptics**. Vukovar: Intech, 2010.

PHIDGETS 2011**. Datasheet: Micro load cell (0-5kg) - CZL635**. Disponível em: <www.phidgets.com>. Acesso em: 7 jun. 2014.

POV RAY. **Free Software tool for creating three-dimensional graphics**. Disponível em: <www.povray.org>. Acesso em: 9 jun. 2014.

ROBLES, G. Virtual Reality: Touch / Haptics. **Encyclopedia of Perception**, California, v. 2, p.1036-1038, 2009.

TAVAKOLI, M.; PATEL, R. V.; MOALLEM, M.; AZIMINEJAD, A. Haptics for teleoperated surgical robot systems. **New Frontiers in Robotics**, v. 1, p. 1-5, 2007.

TEXAS INSTRUMENTS 1996. **Datasheet: OP07C, OP07D, OP07Y Precision operational amplifiers**. Disponível em: <www.ti.com>. Acesso em: 7 jun. 2014.

WEBSTER, G. **Strong Robotic Arm Extends From Next Mars Rover**. Disponível em: <www.jpl.nasa.gov/news/news.cfm?release=2010-301>. Acesso em: 1 out. 2010.

WIRING. **An open-source programming framework for microcontrollers**. Disponível em <wiring.org.co>. Acesso em: 7 jun. 2014.

ZADEH, Mehrdad Hosseini. **Advances in Haptics**. Vukovar: Intech, 2010. 732 p. Disponível em: <http://www.intechopen.com/books/advances-in-haptics>. Acesso em: 31 jul. 2014.

APÊNDICE A – FIRMWARE

```
/*
 * Programa para protótipo da Luva
 * Tui Alexandre Ono Baraniuk
 */

#include <Stepper.h>
#include <Servo.h>
#include <TimedAction.h>

struct MyServo{
   short minPWM;        // min value for servos PWM
   short maxPWM;        // max value for servos PWM
   short minPos;        // min pos value for servos
   short maxPos;        // max pos value for servos
   short pos;           // servos position
   short pwmPin;        // servos PWM pin
};

struct MyStepper{
  int posAux;           // stepper position control
  int maxRotation;      // maximum rotation
  int stepSpeed;        // RPMs Speed
};

struct MySensor{
  int pin;                  // in which AnalogPin the sensor is
connected
  int reading;              // sensor reading
};

struct Glove{
  int mlcToMove;            // min mlc pressure value to move
stepper
  int mlcStaticValue[2];    // mlc value when not moving or under
pressure
  int mlcFFAux;             // mlc forcefeedback aux
  int minContactDefault;    // min fsr value for contact aux
  int fsrMinToClose;        // min pressure value to close the
gloves
```

```
        int fsrMinContact[2];     // min pressure to open gloves
        int previousTime[3];      // time of last movement, 2 for servos
and 1 for stepper
        int lastMovement[3];      // previous movement, {-1, 0, 1}//
close, stop, open// left, stop, right
      };

      //*************************************************/
      // Global Variables
      //*************************************************/
      /*
      Sensor Atributes {sensorPin, reading}
      */
      MySensor fsrSens[4] =  {{2, 0}, {3, 0}, {4, 0}, {5, 0}};
      MySensor mlcSens[2] = {{0, 0}, {1, 0}};

      /*
      Servo Atributes {minPWM, maxPWM, minPos, maxPos, pos, pwmPin,}
      */
      Servo sControl[4]; // 0 and 1: glove, 2 and 3: robots
      MyServo servo[] = {{544, 2350, 0, 40, 0, 2},
                         {544, 2350, 0, 40, 0, 3},
                         {544, 2350, 0, 40, 0, 4},
                         {544, 2350, 0, 40, 0, 5}};

      /*
      Stepper
      Control: {steps, stepPins[4]}
      Atributes: {posAux, maxRotation, stepSpeed}
      */
      Stepper stControl[] = {Stepper (200, 10, 11, 12, 13), Stepper (200,
6, 7, 8, 9)};
      MyStepper stepper[] = {{0, 100, 12},
                             { 0, 100, 12}};
      /*
      Glove Atributes
      {mlcToMove, mlcStaticValue[2], mlcFFAux, minContactDefault,
fsrMinToClose, fsrMinContact[2], previousTime[], lastMovement[3]}
      */
      Glove glove = {15, {520, 465}, 0, 1, 40, {1,1}, {0, 0, 0}, {0,0,0}};
```

```
	/*
	 Thread control
	*/
	static int stepInterval = 50;	// interval before stepper pos is
updated
	static int sensInterval = 1;	// interval before readings
	static int servoInterval = 10;	// interval before servo pos is
updated
	TimedAction stepperThread = TimedAction(stepInterval,
updateSteppers);
	TimedAction sensorsThread = TimedAction(sensInterval,
refreshSensors);
	TimedAction servosThread = TimedAction(servoInterval, updateServos);
	static int spInterval = 50;	// interval before changing
directions
	TimedAction semaphore [3] [2] = {
	 {TimedAction(spInterval, spServoA_open),TimedAction(spInterval,
spServoA_close)},
	 {TimedAction(spInterval, spServoB_open),TimedAction(spInterval,
spServoB_close)},
	 {TimedAction(4*spInterval,
spStepper_left),TimedAction(4*spInterval, spStepper_right)}
	};
	boolean servoA_opening, servoB_opening, servoA_closing,
servoB_closing, st_movingLeft, st_movingRight; // changing directions flags

	//************************************************/
	// Control
	// Initialize Components
	//************************************************/
	void setup(){
	  initServos();
	  initSteppers();
	}

	void loop(){
	  // Thread 1
	  sensorsThread.check();
	  // Thread 2
	  servosThread.check();
	  // Thread 3
```

```
  stepperThread.check();
  // Thread 4
  semaphore[0][0].check(); semaphore[0][1].check();
  semaphore[1][0].check(); semaphore[1][1].check();
  semaphore[2][0].check(); semaphore[2][1].check();
}

//*************************************************/
// Semaphore
//*************************************************/
void spServoA_open(){
  servoA_opening = false;
  semaphore[0][0].disable();
}
void spServoA_close(){
  servoA_closing = false;
  semaphore[0][1].disable();
}
void spServoB_open(){
  servoB_opening = false;
  semaphore[1][0].disable();
}
void spServoB_close(){
  servoB_closing = false;
  semaphore[1][1].disable();
}
void spStepper_left(){
  st_movingLeft = false;
  semaphore[2][0].disable();
}
void spStepper_right(){
  st_movingRight = false;
  semaphore[2][1].disable();
}

//*************************************************/
// Gloves Logic
//*************************************************/
void updateServos(){
  // Test contact and update Servos position
  int currentTime = millis();
```

```
          for(int i = 0; i < 2; i ++){
            // Contact test
            if((fsrSens[i].reading < glove.fsrMinContact[i])){
              if((i == 0 && !servoA_closing)||((i == 1 &&
!servoB_closing))){// check flags
                openServo(i);    // search for contact
                openServo(i+2);  // open robots claw
                if(i == 0){
                  semaphore[i][0].reset(); semaphore[i][0].enable();
                  servoA_opening = true;
                }else{
                  semaphore[i][0].reset(); semaphore[i][0].enable();
                  servoB_opening = true;
                }
              }
            } else{
               // if fsr read is higher than min value and robot sensor read
and semaphore flags ok, then close servo
               if(((fsrSens[i].reading > glove.fsrMinToClose)
               && (fsrSens[i].reading > fsrSens[i+2].reading)))
               {
                 if((i == 0 && !servoA_opening)||((i == 1 &&
!servoB_opening))){// check flags
                   if(!colisionClose()){
                     closeServo(i);    // close gloves claw
                     closeServo(i+2);  // close robots claw
                     if(i == 0){
                       semaphore[i][1].reset(); semaphore[i][1].enable();
                       servoA_closing = true;
                     }else{
                       semaphore[i][1].reset(); semaphore[i][1].enable();
                       servoB_closing = true;
                     }
                   }
                 }
               }
            }
          }
        }

        void updateSteppers(){
```

```
        // Sensors Reading: User rotLeft: higher reading; rotRight: lower
reading; var +- 15
        if((mlcSens[0].reading + glove.mlcFFAux > glove.mlcStaticValue[0]
+ glove.mlcToMove)&& stepper[0].posAux > -stepper[0].maxRotation ){
          // Rotate Left
          if(!st_movingRight){// check flags
            stControl[0].step(-1);
            stepper[0].posAux--;
            stControl[1].step(-1);
            stepper[1].posAux--;
            semaphore[2][0].reset(); semaphore[2][0].enable();
            st_movingLeft = true;
          }
        }else{
          if((mlcSens[0].reading  + glove.mlcFFAux <
glove.mlcStaticValue[0] - glove.mlcToMove)&& stepper[0].posAux <
stepper[0].maxRotation){
            // Rotate Right
            if(!st_movingLeft){// check flags
              stControl[0].step(1);
              stepper[0].posAux++;
              stControl[1].step(1);
              stepper[1].posAux++;
              semaphore[2][1].reset(); semaphore[2][1].enable();
              st_movingRight = true;
            }
          }// else dont move
        }
      }

      // Verify if there is pressure against Robot claw and update
MinContact Variables
      void refreshMinContact(){
        // fsr resistance feedback
        for(int i = 0; i < 2; i++){
          if(fsrSens[i+2].reading - 150 > glove.minContactDefault){ // if
there is force in remote FSRs, update gloves contact variable for force
feedback - 80 for calibration
            glove.fsrMinContact[i] = fsrSens[i+2].reading;
          }else{
```

```
            glove.fsrMinContact[i] = glove.minContactDefault;  // else,
reset to min value
          }
        }

        // mlc forcefeedback
        if(mlcSens[1].reading > glove.mlcStaticValue[1] +
glove.mlcToMove){
          glove.mlcFFAux = mlcSens[1].reading - (glove.mlcStaticValue[1] +
glove.mlcToMove);
        }else{
          if(mlcSens[1].reading < glove.mlcStaticValue[1] -
glove.mlcToMove){
            glove.mlcFFAux = mlcSens[1].reading - (glove.mlcStaticValue[1]
- glove.mlcToMove);
          }else{
            glove.mlcFFAux = 0;
          }
        }
      }

      // Check colision, max permitted position is parallel claws
      bool colisionClose(){
        if((servo[0].pos + servo[1].pos)>(servo[0].minPos +
servo[0].maxPos)){
          return true;
        }else{
          return false;
        }
      }

      //*************************************************/
      // Sensors Logic
      //*************************************************/
      void refreshSensors(){
        for(int i = 0; i < 4; i++){
          fsrSens[i].reading = analogRead(fsrSens[i].pin);
        }
        for(int i = 0; i < 2; i++){
          mlcSens[i].reading = analogRead(mlcSens[i].pin);
        }
```

```
    calibrateFSR();
    refreshMinContact();
  }

  void calibrateFSR(){// Calibrate sensors values based on previous
tests
    fsrSens[0].reading = map(fsrSens[0].reading, 0, 354, 0, 300);
    fsrSens[1].reading = map(fsrSens[1].reading, 0, 337, 0, 300);
    fsrSens[2].reading = map(fsrSens[2].reading, 0, 300, 0, 300);//384
    fsrSens[3].reading = map(fsrSens[3].reading, 0, 390, 0, 300);//413
  }

  //*************************************************/
  // Servos Logic
  //*************************************************/
  void initServos(){
    for(int i = 0; i < 4; i ++){
      sControl[i].attach(servo[i].pwmPin, servo[i].minPWM,
servo[i].maxPWM);
      sControl[i].write(servo[i].pos); // Put servos on initial
position
    }
  }

  void refreshServoPos(int numServo){
    sControl[numServo].write(servo[numServo].pos);
  }

  void closeServo(int numServo){
    if(servo[numServo].pos+1 < servo[numServo].maxPos){
      servo[numServo].pos+=1;
    }else{
      servo[numServo].pos = servo[numServo].maxPos;
    }
    refreshServoPos(numServo);
  }

  void openServo(int numServo){
    if(servo[numServo].pos-1 > servo[numServo].minPos){
      servo[numServo].pos-=1;
    }else{
```

```
      servo[numServo].pos = servo[numServo].minPos;
    }
    refreshServoPos(numServo);
  }

  //*****************************************************/
  // Steppers Logic
  //*****************************************************/
  void initSteppers(){
    for(int i = 0; i < 2; i++){
      stControl[i].setSpeed(stepper[i].stepSpeed);
    }
}
```

APÊNDICE B – PLACA DE CIRCUITO IMPRESSO