

UNIVERSIDADE FEDERAL DO MARANHÃO  
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS  
DEPARTAMENTO DE ENGENHARIA ELÉTRICA  
GRADUAÇÃO EM ENGENHARIA ELÉTRICA

WALBER LIMA PINTO JUNIOR

**ROBÔ CONTROLADO POR GESTOS COM *FEEDBACK* TÁCTIL**

SÃO LUÍS

2017

WALBER LIMA PINTO JUNIOR

**ROBÔ CONTROLADO POR GESTOS COM *FEEDBACK* TÁCTIL**

Monografia apresentada ao Curso de Engenharia Elétrica da Universidade Federal do Maranhão como requisito para obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Luciano Buonocore

SÃO LUÍS

2017

Pinto Junior, Walber Lima.

Robô controlado por gestos com feedback tátil/ Walber Lima Pinto Junior. – 2017.

66 f.

Orientador(a): Luciano Buonocore.

Monografia (Graduação) - Curso de Engenharia Elétrica,  
Universidade Federal do Maranhão, São Luís, 2017.

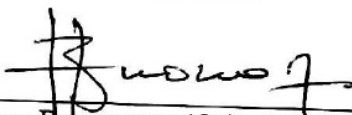
1. Controle por gestos. 2. Feedback tátil. 3. Robótica móvel. 4.  
Comunicação Bluetooth. I. Buonocore, Luciano. II. Título.

WALBER LIMA PINTO JUNIOR

**ROBÔ CONTROLADO POR GESTOS COM *FEEDBACK* TÁCTIL**

Aprovado em: 21 / 07 / 2017

BANCA EXAMINADORA



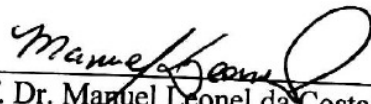
---

Prof. Dr. Luciano Eulonocore (Orientador)  
Departamento de Engenharia Elétrica - UFMA



---

Prof. Dr. Areolino de Almeida Neto (Membro)  
Departamento de Informática - UFMA



---

Prof. Dr. Manuel Leonel da Costa Neto (Membro)  
Departamento de Engenharia Elétrica - UFMA

## **AGRADECIMENTOS**

Agradeço aos meus pais, por todo o apoio e força, sempre me guiando pelo melhor caminho. Obrigado por criarem as bases para a pessoa que sou hoje, prometo retribuí-los o máximo que eu puder.

Ao meu orientador Luciano Buonocore, por toda a assistência prestada, mais até do que o necessário, tornando a realização deste trabalho muito mais tranquila. Agradeço-o profundamente.

Ao meu amigo Deocarolo Guinzani, pela grande ajuda na confecção da luva utilizada para este trabalho.

Aos amigos do LRC, em especial Daniel Ribeiro e Luís Eugênio, que me ajudaram imensamente durante tanto a parte prática quanto teórica deste trabalho. Sem o conhecimento compartilhado por vocês, este trabalho não seria o mesmo.

## RESUMO

A proposta implementada neste trabalho de monografia é realizar o controle dos movimentos de um robô de baixo custo por gestos manuais. A comunicação entre os módulos do controlador manual e o módulo embarcado no robô (executando em uma placa Arduino) faz-se através do sistema *Bluetooth*, amplamente difundido. Para que o operador do sistema mecatrônico tenha condições de avaliar possíveis colisões do robô com obstáculos a sua volta, foi empregada a técnica de *feedback* tátil que, de forma indireta, fornece informações de intervalos das distâncias frontal e lateral aos obstáculos. Com isso, o operador poderá tomar decisões dos movimentos seguintes ou mesmo parar o robô. Foram feitos testes de repetibilidade nos sistemas isolados, em conjunto e de forma integrada, envolvendo todo o sistema robótico (controlador manual com *feedback* tátil e robô) com o objetivo de validar a robustez do sistema proposto no trabalho. Em todos os testes realizados, sem exceção, as operações de controle de movimento dos robôs ocorreram no sentido comandado e da monitoração através dos motores de vibração (*feedback* tátil) dentro dos valores esperados, medidos indiretamente através da tensão nos terminais dos motores. Ainda que apresente um alcance pequeno em comunicação, limitado pelo sistema *Bluetooth*, restrições a movimentos (isolados ou compostos) em velocidades constantes e seja utilizado um robô de baixo custo, o sistema implementado neste trabalho representa a validação de uma prova conceitual de que a teleoperação com *feedback*, em especial o tátil, é uma possibilidade presente e já utilizada em diversas áreas robóticas.

**Palavras-chaves:** Controle por gestos; *Feedback* tátil; Robótica móvel; Comunicação *Bluetooth*.

## ABSTRACT

The proposal implemented in this monograph is to control the movements of a low-cost robot by manual gestures. Communication between the modules of the manual controller and the embedded system in the robot (running on an Arduino board) is done through the Bluetooth system, now widely used. In order for the mechatronic system operator to be able to evaluate possible collisions of the robot with obstacles around it, a tactile feedback technique was used, which indirectly provides information about intervals of frontal and side distances to obstacles. With this, the operator can make decisions of the following movements or even stop the robot. Repeatability tests were performed on isolated devices, partial systems and in an integrated way, involving the whole robotic system (manual controller with tactile feedback and robot) in order to validate the robustness of the system proposed in this paper work. In all tests carried out, without exception, the movement control operations of the robots occurred in the commanded direction and the monitoring through the vibration motors (tactile feedback) within the expected values, measured indirectly through the voltage at the terminals of the motors. Even if it presents a small range in communication, limited by the Bluetooth system, restrictions to movements (isolated or compound) at constant speeds and a low cost robot is used, the system implemented in this work represents the validation of a conceptual proof that the teleoperation with feedback, especially the tactile, is an actual possibility and already used in several robotic areas.

**Keywords:** Control by gestures; Tactile feedback; Mobile robotics; Bluetooth communication.

## LISTA DE FIGURAS

Figura 1	Máquinas na Revolução Industrial.....	14
Figura 2	Braço robótico em processo industrial.....	15
Figura 3	Drone sendo utilizado em irrigação de produtos agrícolas.....	15
Figura 4	Drone sendo utilizado em vigilância de fronteiras.....	16
Figura 5	Tarefas em robótica móvel.....	17
Figura 6	Braço mecânico teleoperado.....	18
Figura 7	a) Robô com sensor de distância b) Robô seguidor de linha com câmera.....	20
Figura 8	Deficiente visual utilizando óculos com sensor de distância.....	20
Figura 9	Ilustração de celular vibrando.....	21
Figura 10	Joystick com função de <i>feedback</i> tátil.....	22
Figura 11	Foto do robô utilizado.....	24
Figura 12	Placa Arduino Uno.....	26
Figura 13	Módulo <i>Bluetooth</i> HC-05.....	26
Figura 14	Sensores: a) infravermelho; e b) ultrassônico.....	27
Figura 15	Foto da placa de interface utilizada.....	28
Figura 16	Placa MD-25.....	28
Figura 17	Seleção da placa utilizada na IDE.....	30
Figura 18	Diagrama de blocos dos dois sistemas em conjunto.....	31
Figura 19	Diagrama de blocos do código de controle do robô móvel.....	31
Figura 20	Placa Pro Mini 328p a) frente e b) costa.....	34
Figura 21	Motor de vibração 1027.....	34
Figura 22	a) Módulo MPU-6050 b) Eixos do módulo.....	35
Figura 23	Projeto de circuito do sistema do controlador manual.....	36
Figura 24	Protótipo de circuito para o sistema de controlador manual com <i>feedback</i> tátil.....	36
Figura 25	Placa universal utilizada.....	37
Figura 26	Placa de circuito do controlador manual.....	38
Figura 27	a) Sistema do controlador manual com <i>feedback</i> tátil b) sistema sendo utilizado por um operador.....	39
Figura 28	Diagrama de blocos da programação do controlador manual.....	40
Figura 29	Ligação entre o Arduino Uno e o módulo HC-05.....	43



## LISTA DE TABELAS

Tabela 1	Movimentos do robô relacionados aos valores codificados do acelerômetro....	32
Tabela 2	Intensidade da vibração dos motores relacionada aos valores codificados das distâncias lidas.....	40
Tabela 3	Análise dos dados coletados do acelerômetro para o eixo X.....	44
Tabela 4	Análise de medições do infravermelho.....	45
Tabela 5	Análise de medições do sonar direito.....	45
Tabela 6	Análise de medições do sonar esquerdo.....	45
Tabela 7	Relação entre tensão média nos motores e <i>Duty Cycle</i> utilizado.....	47

## LISTA DE ABREVIATURAS E SIGLAS

SLAM	<i>Simultaneous Localization and Mapping</i>
TEEE	Tópicos Especiais de Engenharia Elétrica
ROTEX	<i>Robot Technology Experiment</i>
LED	<i>Light Emitter Diode</i>
RoSeLi	Robô Seguidor de Linha
LRC	Laboratório de Robótica Móvel e Comunicação Sem Fio
I2C	<i>Inter Integrated Circuit</i>
USB	<i>Universal Serial Bus</i>
IDE	<i>Integrated Development Environment</i>
UART	<i>Universal Asynchronous Receiver Transmitter</i>
CPU	<i>Central Processing Unit</i>
MIPS	<i>Millions of Instructions Per Second</i>
RAM	<i>Random-Access Memory</i>
EEPROM	<i>Electrically Erasable Programmable Read-Only Memory</i>
SPI	<i>Serial Peripheral Interface</i>
PWM	<i>Pulse Width Modulation</i>

## SUMÁRIO

1	INTRODUÇÃO .....	11
2	FUNDAMENTAÇÃO TEÓRICA .....	14
2.1	Tarefas comuns em robótica móvel .....	16
2.2	Robótica teleoperada.....	18
2.3	Realimentação sensorial humana .....	19
2.4	Realimentação tátil.....	21
3	DESCRIÇÃO DO ROBÔ RoSeLi.....	24
3.1	Parte eletrônica.....	25
3.2	Parte computacional.....	29
4	IMPLEMENTAÇÃO DO SISTEMA DE COMANDO DO ROBÔ NA LUVA.....	33
4.1	Componentes utilizados .....	33
4.2	Placa de circuito do sistema de comando.....	35
4.3	Acoplamento do sistema de comando à luva .....	37
4.4	Parte computacional.....	37
5	EXPERIMENTOS E RESULTADOS .....	42
5.1	Comunicação entre dispositivos <i>Bluetooth</i> .....	42
5.2	Acelerômetro.....	43
5.3	Sensores de distância .....	44
5.4	Envio de dados do acelerômetro via <i>Bluetooth</i> .....	46
5.5	Acionamento de motores de vibração .....	46
5.6	Acionamento de motores do robô móvel .....	47
5.7	Controle de movimento do robô .....	48
5.8	<i>Feedback</i> tátil.....	48
5.9	Sistema completo .....	49
6	CONCLUSÕES E TRABALHOS FUTUROS .....	50
	REFERÊNCIAS BIBLIOGRÁFICAS .....	51
	APÊNDICE A – Códigos em C/C++ no ATmega328p para realização dos experimentos em componentes e sistema completo .....	54

## 1 INTRODUÇÃO

A robotização de processos nas mais diversas áreas humanas já é uma realidade no mundo moderno. Nas indústrias os robôs móveis, articulados ou híbridos (móvel com capacidade de manipulação de objetos) são utilizados largamente, de forma a otimizar os processos de produção, montagens, etc. A gama de aplicações dessas máquinas consiste desde um baixo nível de inteligência, porém executando tarefas repetitivas (braços robóticos empregados em uma linha de montagem de automóveis) até as empregadas em atividades de produção que requerem um maior grau de inteligência (devem tomar decisões em função do estado atual do processo).

A robótica móvel em particular é bastante pesquisada na realização das mais diversas tarefas como localização, exploração, SLAM (*Simultaneous Localization And Mapping* – Localização e Mapeamento Simultâneos), atividades que normalmente pedem inteligência da plataforma móvel. Entretanto, existem casos onde essas atividades devem ser comandadas por um operador humano, como são os casos dos robôs teleoperados. Nesse tipo de aplicação, o sistema mecatrônico obedece aos comandos de um usuário, que assim controla os movimentos de um robô, ou mesmo realiza a manipulação de objetos, no caso de robôs híbridos. Existem também aplicações que envolvem a teleoperação de braços robóticos, como os utilizados em cirurgias à distância.

Esse trabalho de monografia aborda o comando de robô teleoperado com realimentação (*feedback*) tátil de informações, possibilitando ao usuário do sistema tomar decisões em função desses dados recebidos.

### 1.1 Objetivos

O objetivo geral deste trabalho de monografia é implementar um sistema de controle dos movimentos de um robô utilizando *feedback* tátil.

Com relação aos objetivos específicos, este trabalho propõe:

- a) Controlar os movimentos do robô em oito possibilidades de sentidos, além de comando de parada, de forma individual (frente, ré, giro horário ou anti-horário) ou combinada (frente à direita, ré à esquerda, etc.) utilizando gestos manuais;

- b) Fazer a comunicação entre o sistema de controle, localizado na mão do usuário, com o robô móvel através do padrão *Bluetooth*, em ambos os sentidos, enviando comandos de movimento e recebendo realimentação de distâncias na forma tátil; e
- c) Mostrar a simplicidade da eletrônica envolvida em todo o sistema (controle manual com *feedback* tátil e robô móvel de baixo custo).

## 1.2 Motivação

As motivações para a escolha deste tema devem-se a alguns fatores. O primeiro e principal é a proposta de baixo custo envolvida nos comandos de uma plataforma móvel de forma precisa e visual. O aspecto da precisão está relacionado à possibilidade de ajustar os movimentos do robô de acordo com os gestos manuais suaves na direção desejada. O aspecto visual pode ocorrer tanto no campo de visão do operador ou por imagens do local onde o mesmo está sendo deslocado.

Essa motivação, de forma isolada, carece de informações adicionais de distâncias do robô aos objetos em sua vizinhança. Nem sempre a informação visual *in loco* ou por imagem possibilita uma forma adequada de mensurar essas distâncias. Dessa forma, como segunda motivação existe a possibilidade de *feedback* tátil, onde as distâncias mencionadas são mensuradas por sensores *onboard* e retornadas na forma de vibração (efeito *tátil* produzido na mão) que melhor instrui o operador do sistema sobre os movimentos seguintes.

Por fim, ainda existe a contribuição da implementação e validação desta proposta de trabalho na formação acadêmica, aprimorando os conhecimentos adquiridos na disciplina TEEE – Robótica Móvel Probabilística cursada durante a graduação. Além disso, esse trabalho pode servir como material pedagógico didático-prático da referida disciplina, exemplificando sistemas teleoperados com realimentação.

## 1.3 Organização do trabalho

O trabalho está organizado em seis capítulos, incluindo este de introdução do tema.

No Capítulo 2 são abordados os temas relacionados aos elementos envolvidos na implementação do trabalho, mais especificamente a relação entre a autonomia dos robôs e sua teleoperação, bem como as diferentes formas que se pode apresentar a realimentação nesses sistemas, com ênfase na forma tátil.

No Capítulo 3 é apresentado o robô utilizado no trabalho, modificado de uma aplicação anterior, de forma a se adequar nos elementos de *hardware* e *software* às necessidades deste trabalho. O módulo de *software* embarcado na placa Arduino é apresentado em alto nível e os detalhes do código no Apêndice A.

O Capítulo 4 é dedicado ao detalhamento do projeto e implementação do *hardware* e *software* do controlador manual, que deverá ser posicionado através de uma luva inserida na mão do operador. Novamente, o módulo de *software* é descrito em alto nível (diagrama de blocos) e seu detalhamento em código é disponibilizado no Apêndice A.

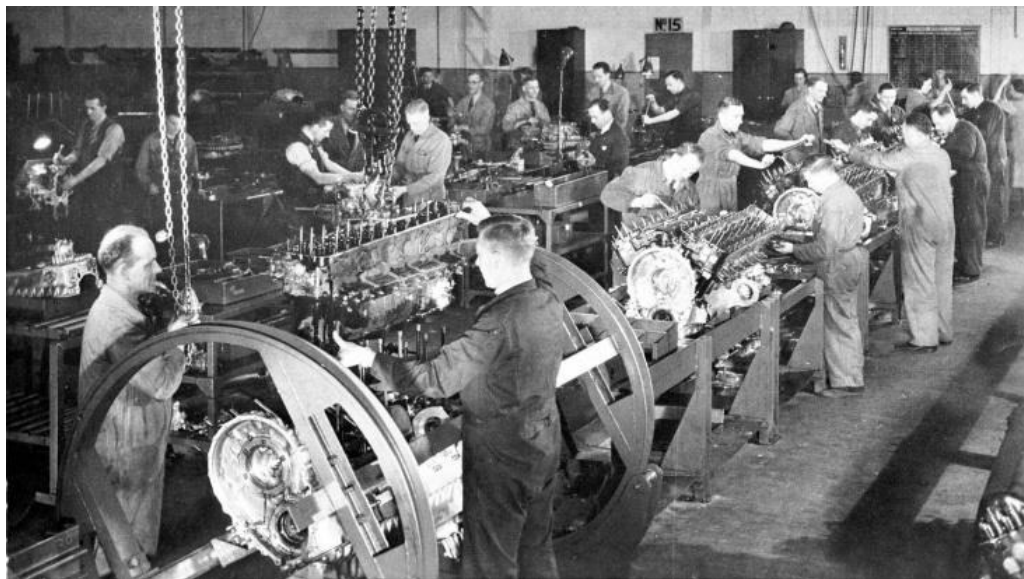
No Capítulo 5 são realizados os experimentos nos componentes isolados e em conjunto, sendo o último experimento feito no sistema completo. Os testes realizados tiveram o objetivo de comprovar a robustez do sistema (individualmente, em conjunto e totalmente integrado), nas ações comandadas e recebidas como realimentação, fazendo-se uma quantidade de testes repetitivos.

Finalmente, no Capítulo 6 apresentam-se as conclusões e os trabalhos futuros que podem ser realizados a partir do atual estado obtido neste trabalho.

## 2 FUNDAMENTAÇÃO TEÓRICA

A robótica é uma das áreas de pesquisa sobre tecnologia que vem sendo amplamente desenvolvida nas últimas quatro décadas. A utilização de máquinas que operam autonomamente, realizando tarefas muitas vezes repetitivas e cansativas para o ser humano, já é realidade desde a época da revolução industrial (Figura 1), porém a robótica em si é um segmento mais recente, onde são utilizadas formas mais complexas e inteligentes de automação dos processos.

Figura 1 – Máquinas na Revolução Industrial



Fonte: <http://www.historiadetudo.com/revolucao-industrial>

As aplicações da robótica são extensas, como em atividades de laser e entretenimento, no transporte de materiais, em cirurgias médicas e na exploração espacial. Uma destas aplicações encontra-se na área da educação, onde se ensina os conceitos básicos, tanto teóricos quanto práticos, sobre robótica aos alunos, com o intuito de estimular a criatividade, a multidisciplinaridade e o raciocínio lógico dos alunos (ZILLI, 2004). A sua integração dentro das indústrias pode ser feita por meio de braços robóticos, por exemplo, onde estes têm a função de selecionar certas peças, fazendo seus movimentos, como em um processo de montagem (Figura 2) ou operando em áreas de risco e de difícil acesso ao ser humano (RIBEIRO, 2013).

Figura 2 – Braço robótico em processo industrial



Fonte: <http://roboticacolegiovida.blogspot.com.br/2014/04/braco-mecanic.html>

Outro exemplo interessante a ser comentado é o uso de drones, um tipo de veículo aéreo não tripulado, em diferentes aplicações. Os drones são robôs que se utilizam de hélices para voar e podem ser utilizados em diferentes tarefas, como irrigação e colheita de produtos agrícolas (Figura 3) ou em questões de segurança nacional, como na vigilância de fronteiras entre territórios de diferentes países (SANTOS Jr., 2016), conforme mostrado na Figura 4.

Figura 3 – Drone sendo utilizado em irrigação de produtos agrícolas



Fonte: [http://blog.droneng.com.br/dji\\_agras\\_m1/](http://blog.droneng.com.br/dji_agras_m1/)



Figura 4 – Drone sendo utilizado em vigilância de fronteiras



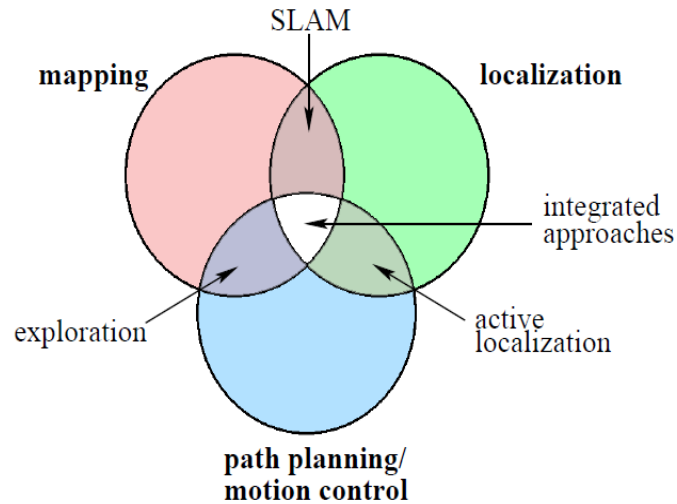
Fonte: <http://devdronsehop.in/blog/tipos-uso-los-drones/>

## 2.1 Tarefas comuns em robótica móvel

A robótica móvel, como o próprio nome sugere, trata da seção da robótica onde se opera com robôs que possuem algum tipo de sistema de locomoção, como rodas, esteiras ou pernas. Existem diversas tarefas que podem ser executadas por robôs móveis, normalmente se referindo a problemas a serem resolvidos pelo sistema, entre os básicos pode-se citar: localização, mapeamento e planejamento de trajetória, conforme ilustrado na Figura 5. Dessas três tarefas básicas, surgem outras que representam a combinação de duas ou até todas as tarefas, este último sendo o caso do problema SLAM autônomo ou abordagem integrada. Nessa tarefa o robô deve realizar a localização e o mapeamento do ambiente de forma simultânea, sendo ambas as informações desconhecidas (BUONOCORE, 2013).

A localização consiste em o robô descobrir sua pose (posição e orientação) em um mapa pré-definido, ou seja, conhecido *a priori*. A partir do uso de diferentes sensores, o robô consegue, após tomar várias medições, estimar sua pose dentro do mapa. Sabendo a sua localização, pode-se, por exemplo, realizar alguma tarefa como navegar para um local específico para realizar alguma operação.

Figura 5 – Tarefas em robótica móvel



Fonte: (STACHNISS, 2006)

Durante os movimentos realizados pelo robô no ambiente, ocorrem erros de odometria que representam as diferenças entre os movimentos efetivamente realizados e os informados por sensores. Esses erros tendem a ser acumulados, caso não exista um sistema de filtragem que os minimiza em função das medições de distâncias feitas pelo robô aos objetos do ambiente. Algumas das causas desse tipo de erro são a utilização de rodas de diâmetro diferente, projeto inadequado do centro de carga da estrutura do robô e movimentação em superfície não uniforme (rugosa ou molhada, por exemplo), causando alterações na trajetória que são difíceis de serem mensuradas<sup>1</sup>.

Na tarefa de mapeamento, o robô se locomove por um local desconhecido, tomando medidas de obstáculos ou paredes, porém tendo suas poses conhecidas *a priori*. As poses do robô poderão ser fornecidas por algum sistema externo de localização ou mesmo de forma manual. A função dessa tarefa é verificar o sistema de medição do robô, já que se assume erro de odometria nulo (poses conhecidas).

A última tarefa é o planejamento de trajetória, onde o robô conhece o mapa e sua pose inicial. Nesta tarefa, é dado um objetivo ao robô (um local no mapa onde ele deve chegar), e cabe ao mesmo calcular a melhor trajetória para chegar neste local. Vale ressaltar que a “melhor” trajetória é algo relativo, podendo significar menor tempo de realização do percurso,

<sup>1</sup> Site: <http://users.isr.ist.utl.pt/~mir/cadeiras/robmovel/Odometry.pdf>

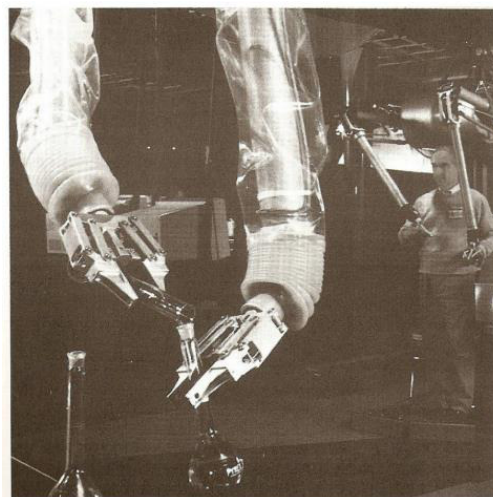
menor gasto de recursos (como energia do robô) ou menor incerteza na realização da tarefa (THRUN *et al.*, 2006).

## 2.2 Robótica teleoperada

As aplicações em robótica teleoperada são definidas por tarefas onde o robô é controlado a distância por um usuário (não autônomo), sem o uso de fios na comunicação entre ambos. Esta comunicação pode ser feita utilizando diferentes métodos e tecnologias existentes, variando em custo, alcance, velocidade, etc. Podemos citar como alguns exemplos as comunicações via infravermelho, radiofrequência (como o *Bluetooth*, por exemplo) e *wifi*.

O uso deste tipo de tecnologia se faz necessária principalmente quando o trabalho a ser realizado está relacionado a condições ou ambientes insalubres ao ser humano, como operações em locais radioativos, manipulação de peças em temperaturas extremas, exploração espacial (falta de oxigênio), etc. Além da manipulação a distância, o usuário também pode receber, a depender do sistema, algum tipo de realimentação sensorial, na forma visual, sonora ou tátil, o que lhe permite ter um maior senso e entendimento do processo que está sendo realizado (TOURINO, 1999). A Figura 6 apresenta um braço mecânico com operação a distância.

Figura 6 – Braço mecânico teleoperado



Fonte: (ZHAI, 1992)

Como exemplo de aplicações nesta área, tem-se o robô ROTEX (*Robot Technology Experiment*), enviado ao espaço em 1999 e usado dentro de uma espaçonave para abrir e

fechar conectores, montar estruturas e capturar objetos em flutuação (HIRZINGER, 1994). Também se pode citar o uso da teleoperação em cirurgias médicas, onde o cirurgião pode fazer o procedimento no paciente à distância, de forma remota ao local onde se encontra o atendido (BALLANTYNE, 2002). Nas situações que envolvem a teleoperação, a realimentação de informações ao operador é de fundamental importância na realização da operação em curso. Essa realimentação pode se dar de diversas formas, normalmente chamada de realimentação sensorial humana.

### 2.3 Realimentação sensorial humana

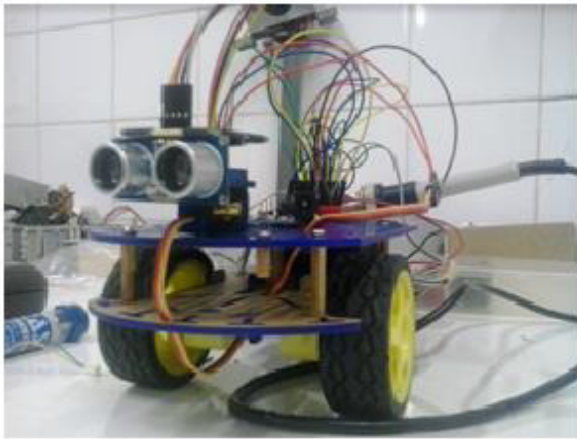
A realimentação pode ser definida como uma resposta a algo que está acontecendo, de forma a prover mais informações sobre uma ação. Na robótica, este conceito aplica-se quando um robô, após realizar alguma ação, envia informações ao seu usuário ou a si mesmo, de forma que essas novas informações são auxiliares ou necessárias na tomada de decisão das próximas ações. Esta nova forma de aquisição e utilização de informações adquirida pelos robôs trouxe a eles a capacidade de sentir e interagir com o ambiente onde estão operando, criando assim uma nova lista de possíveis aplicações à robótica (NUNES, 1995).

Exemplos comuns de realimentação na robótica são os robôs móveis que se utilizam de sensores de distância (permitindo a eles desviar de obstáculos) ou detecção de linhas (seguem linhas do ambiente). No caso do robô seguidor de linha, pode-se fazer uso de processamento de imagem via câmera (RIBEIRO, 2017) ou, simplesmente, sensores infravermelhos para a detecção e alinhamento na linha a ser seguida, conforme ilustra a Figura 7.

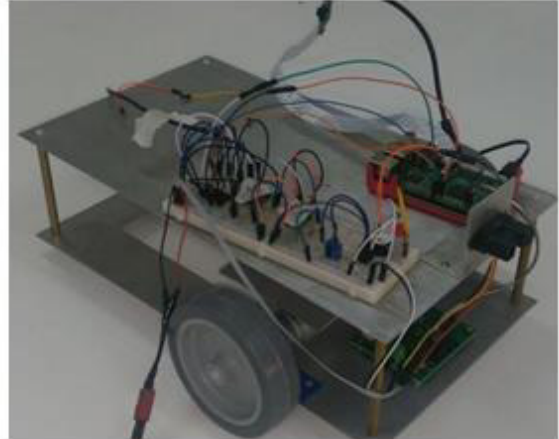
Outro exemplo de aplicação da realimentação é a automação de sistemas de irrigação, onde sensores de umidade captam informações sobre o solo, verificando a necessidade de se aumentar ou diminuir a quantidade de água utilizada no processo. Tal sistema de realimentação, além de reduzir o gasto de recursos, torna o crescimento das plantas mais eficiente e diminui a necessidade da intervenção humana no processo (GUIMARÃES, 2011).

No caso de a realimentação ser enviada ao ser humano que opera um robô, a informação é passada de forma que o operador possa entendê-la, ou seja, por meio dos sentidos humanos. Os casos mais comuns são a realimentação por meio da visão e da audição, onde o robô emite sinais visuais ou sonoros ao seu operador, que captará as informações sobre o robô ou sobre o ambiente onde o mesmo está inserido. Por exemplo, pode-se usar uma sinalização por meio de LEDs, *display*, *buzzer* ou alto-falante para indicar baixa tensão na alimentação do sistema,

Figura 7 – a) Robô com sensor de distância b) Robô seguidor de linha com câmera



(a)



(b)

Fonte: AUTOR

superaquecimento de um circuito impresso, movimentação próxima a uma máquina em operação, etc.

A realimentação sensorial humana por si só já possui aplicações próprias, independente da robótica, como é o caso de sensores de distância sendo usados para informar, por meio de sinais sonoros, a proximidade de objetos a um deficiente visual (GONZATTO et al., 2009) (Figura 8).

Figura 8 – Deficiente visual utilizando óculos com sensores de distância



Fonte: (GONZATTO et al., 2009)

## 2.4 Realimentação táctil

Para entender o que significa a realimentação táctil, primeiramente as duas palavras devem ser interpretadas separadamente. Realimentação, como dito anteriormente, é o processo de devolução de informações, dando uma resposta a respeito de um evento. Assim, a realimentação táctil é uma forma de realimentação que se utiliza de ações que podem ser sentidas pelo corpo humano, como vibração, movimento ou pressão.

O avanço da tecnologia está levando a uma interação mais próxima entre seres humanos e máquinas. Novas características estão sendo adquiridas por dispositivos, provendo diferentes técnicas de transmissão de informações. Ao interagir com dispositivos, experiências de realimentação foram ineficientes ao lidar com telas sensíveis de telefones celulares. Diferentemente de *smartphones* mais antigos, um moderno tem a característica de responder a uma interação do usuário com vibração, conforme ilustra a Figura 9. Isto é devido ao avanço da tecnologia na área da realimentação táctil, dando aos usuários uma experiência melhor em serviço.

Figura 9 – Ilustração de celular vibrando



Fonte: <http://catafau.blogspot.com.br/2013/07/seu-cerebro-faz-o-celular-vibrar.html>

Este tipo de tecnologia pode ser aplicado em diferentes ambientes e dispositivos, além dos telefones celulares. A realimentação táctil também é empregada em muitos *joysticks* de videogames, aumentando os efeitos dados pelo jogo (por exemplo: uma explosão ou tiroteio provocam uma vibração no *joystick*), trazendo mais realismo aos jogadores (Figura 10). Em qualquer caso, para se utilizar um sistema com realimentação táctil, deve-se ter em mente a velocidade da movimentação, a força e a precisão desejada para a aplicação, pois esta

Figura 10 – Joystick com função de *feedback* tátil



Fonte: <http://www.ebay.com/itm/Black-USB-Dual-Vibration-PC-Computer-Wired-Gamepad-Game-Controller-Joystick-New-/350954087368>

informação será repassada diretamente ao ser humano operador do sistema (BARANIUK, 2014).

Estendendo este conceito para a área médica, uma aplicação importante e relativamente nova da realimentação tátil ocorre em cirurgias. O desenvolvimento de máquinas cirúrgicas com esta tecnologia começou nos Estados Unidos, onde tais máquinas foram capazes de prover esta realimentação aos cirurgiões, aumentando seus sentidos tácteis e permitindo-lhes sentir as ferramentas mantidas pela máquina operando a cirurgia. Esse recurso fornece mais segurança e precisão em seus movimentos em procedimentos delicados (PAYNE, 2015).

Robôs móveis também estão começando a utilizar esta realimentação, aumentando seu desempenho onde outros tipos de realimentação não estão presentes ou não são fortes o suficiente para fornecer as informações necessárias. Por exemplo, quando um robô manipulador é obrigado a mover um objeto que se encontra parado, a realimentação a partir da visão não permitirá ao operador saber quanta força o manipulador está aplicando ao objeto (COUNSELL, 2003).

Com o crescente uso e desenvolvimento da robótica, questões éticas e regulamentos importantes de seu uso estão sendo discutidos. Na área médica, por exemplo, robôs autônomos estão sendo desenvolvidos para fazer cirurgias, melhorando a precisão e a eficiência do processo. Por outro lado, começam a surgir questões relativas à responsabilidade de eventuais falhas e problemas. Além do médico e do hospital, o fabricante do robô também

pode assumir a responsabilidade<sup>2</sup>. Mesmo com o aumento da precisão e segurança de alguns processos robóticos, algumas pessoas podem não confiar na máquina na condição de cirurgião e ainda podem preferir um ser humano fazendo este tipo de operação.

Neste trabalho de monografia será utilizada esta forma de realimentação, conhecida como *feedback* tátil. O trabalho implementado consistiu em implementar um controlador manual para um robô móvel, o qual deve realizar o *feedback* tátil para a mão do usuário, informando-o de objetos próximos ao robô, codificado em intervalos de distâncias. O *feedback* tátil é usado não só neste trabalho, mas em diversos outros sistemas, como forma de complementar os dados recebidos por um operador, em acréscimo a outras formas de *feedback*, como visão e audição, com o objetivo de adquirir dados essenciais do sistema.

---

<sup>2</sup> Site: <https://www.ncbi.nlm.nih.gov/pubmed/20224537>

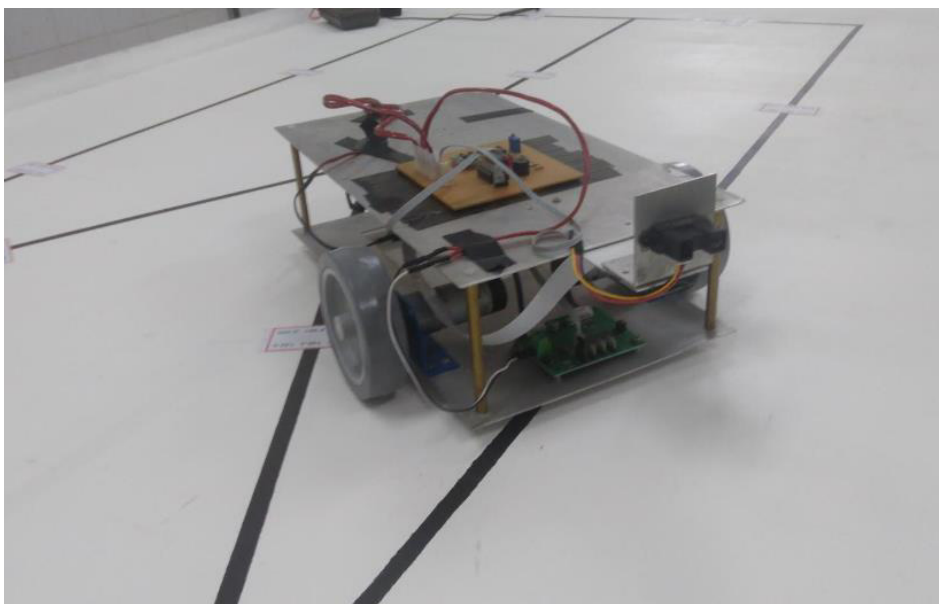


### 3 DESCRIÇÃO DO ROBÔ RoSeLi

Neste capítulo é descrito o circuito eletrônico projetado para o controle do robô. Além disso, também serão apresentados os diagramas em blocos contendo os elementos de ambos os sistemas em conjunto (controlador manual e robô) e a programação necessária para o controle do robô. Apenas as funções executadas pelo código do microcontrolador existente na placa Arduino embarcada no robô é detalhado neste capítulo, enquanto a do controlador manual (presente na mão do usuário e incorporando o *feedback* tátil), doravante definido como controle manual com *feedback* tátil, é apresentado no Capítulo 4.

O robô que será utilizado para este trabalho é denominado RoSeLi (**Robô Seguidor de Linhas**) e foi construído para a realização de outro trabalho de monografia (RIBEIRO, 2017). A base de sustentação do robô é implementada por placas de alumínio e para o controle dos motores é utilizado um sistema de controle (placa MD-25), um microcontrolador (ATmega328P, contido na placa Arduino Uno) e uma placa de interface, esta última projetada e construída (assim como a base mecânica do robô) no LRC (Laboratório de Robótica Móvel e Comunicação Sem Fio), localizado no Centro de Ciências Exatas e Tecnológicas da Universidade Federal do Maranhão. Foram feitas pequenas alterações para adequação a este trabalho. Pode-se visualizar o robô antes de quaisquer modificações na Figura 11.

Figura 11 – Foto do robô utilizado



Fonte: (RIBEIRO, 2017)

Foi adicionado ao robô um sistema contendo um módulo *Bluetooth* e dois sensores ultrassônicos. O primeiro é utilizado para criar a comunicação entre o robô e o controlador manual com *feedback* tátil, doravante referido apenas como controlador manual, o qual está localizado sobre uma luva, vestida na mão do operador. Os sensores ultrassônicos são utilizados para verificar possíveis objetos nas laterais do robô. Estas informações de distâncias do robô aos objetos são enviadas ao operador na forma de vibrações. Essas percepções táteis são criadas a partir de pequenos motores DC (próprios para aplicações de vibração) acoplados no controlador manual.

### 3.1 Parte eletrônica

No que diz respeito à parte mecânica, o robô não teve nenhuma mudança neste trabalho de monografia, somente a substituição do sistema embarcado utilizado no trabalho anterior (Raspberry Pi 3) por um outro mais simples (Arduino Uno) e o acoplamento de sonares nas laterais e do módulo *Bluetooth*. Dessa forma, a parte eletrônica do robô RoSeLi é constituída de três partes:

- a) A placa Arduino Uno (microcontrolador ATmega328P), de onde saem conexões para o módulo *Bluetooth*, um sensor infravermelho (localizado na frente do robô, e já existente) e um par de sensores ultrassônicos, localizados nas laterais (acrescidos).
- b) Uma placa de interface, já implementada anteriormente, para controle de alimentação dos motores e com circuito de detecção de bateria baixa.
- c) A placa MD-25, que efetua o controle da movimentação do robô, comunicando-se com o microcontrolador via protocolo I2C (*Inter Integrated Circuit*), também já existente no robô.

A placa Arduino Uno foi escolhida para este trabalho devido a ela possuir os recursos de *hardware* suficientes para a aplicação a que destina este sistema de controle proposto, ser de baixo custo e disponível no LRC, ser de fácil programação, que se dá via cabo USB (*Universal Serial Bus*) e utilizando IDE (*Integrated Development Environment*) própria (linguagem C/C++), que será comentada mais a frente. Pode-se ver a placa Arduino Uno na Figura 12.

Figura 12 – Placa Arduino Uno



Fonte: <http://www.arduino.org/products/boards/arduino-uno>

O módulo *Bluetooth* utilizado é o HC-05, que pode ser configurado como mestre ou escravo<sup>3</sup> (Figura 13). Foi utilizado um par deste módulo: um para o robô e outro para o controlador manual, onde o módulo do robô foi configurado como escravo, enquanto o montado na luva (controle manual) foi configurado como mestre.

Figura 13 – Módulo *Bluetooth* HC-05



Fonte: <http://artofcircuits.com/product/hc-05-bluetooth-serial-pass-through-master-slave-module>

A diferença entre as configurações de mestre e escravo do módulo *Bluetooth* representa a possibilidade de criar comunicações entre diferentes dispositivos *Bluetooth*: o dispositivo mestre pode criar e estabelecer novas conexões/pareamentos, enquanto o escravo apenas

<sup>3</sup> Site: <http://www.instructables.com/id/How-to-Configure-HC-05-Bluetooth-Module-As-Master-/>

recebe e aceita essa comunicação. Logo, um par de módulos, ambos configurados como escravos, não conseguiriam estabelecer conexão direta entre si.

Os sensores de distância escolhidos são de baixo custo, também disponíveis para uso no laboratório LRC. São eles: sensor de distância infravermelho GP2Y0A41SK0F<sup>4</sup> e sensor de distância ultrassônico HC-SR04<sup>5</sup>, ambos operados a 5 V (Figura 14).

Figura 14 – Sensores: a) infravermelho; e b) ultrassônico



Fonte: <https://www.coolcomponents.co.uk/>

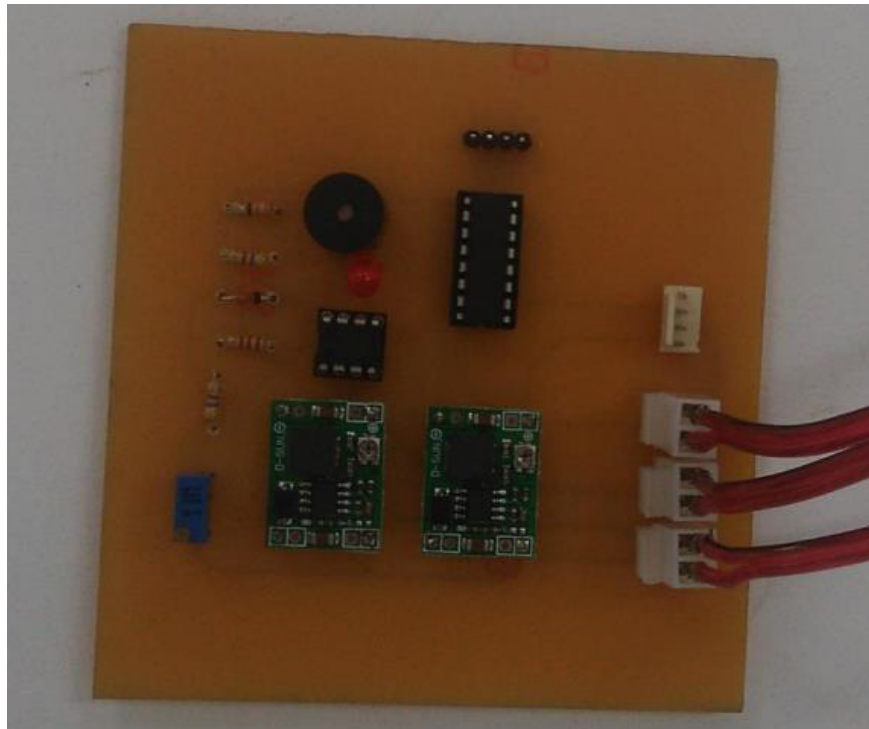
A alimentação do sistema será dada por uma bateria de lítio polímero de 16,8 V de tensão máxima. Para ajustar esta tensão aos níveis utilizados pelos motores (12 V), pelo microcontrolador e pelos sensores (5 V), se faz necessário o uso de dois conversores CC-CC Buck já contidos na placa de interface. Essa placa também incorpora um detector de bateria baixa, que quando alcança um nível considerado crítico, adotado como 14V, emite sinais sonoros e visuais para alertar que a bateria necessita ser recarregada (RIBEIRO, 2015). A foto da placa montada é mostrada na Figura 15.

A placa MD-25 (Figura 16) é um driver de motores, que realiza leitura de *encoders* de motores e tensão/corrente dos mesmos, sendo esses dados transferidos via I2C ou UART (*Universal Asynchronous Receiver Transmitter*) para o controlador do sistema (no caso, o ATmega328P). Os comandos disponíveis para esta placa podem ser encontrados na fonte da Figura 16.

<sup>4</sup> Site: [http://www.sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y0a41sk\\_e.pdf](http://www.sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y0a41sk_e.pdf)

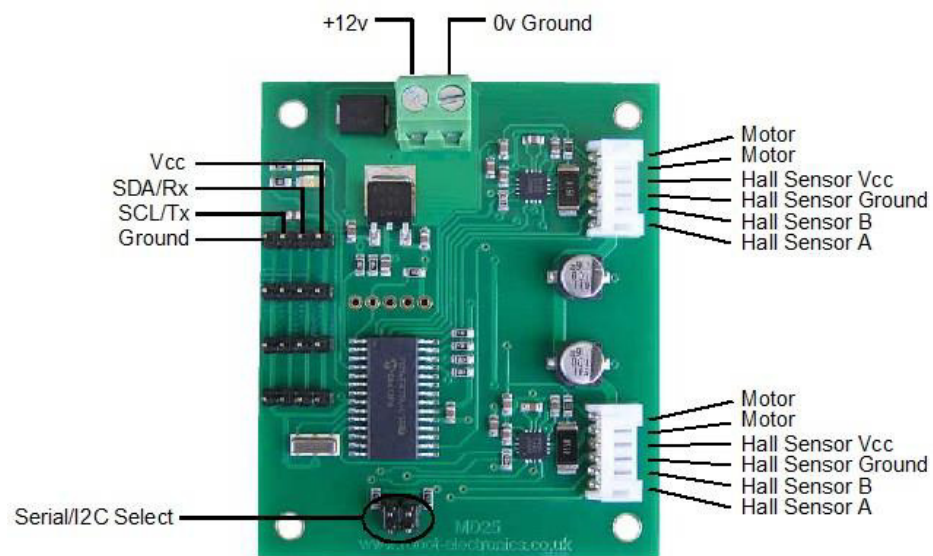
<sup>5</sup> Site: <http://www.micropik.com/PDF/HCSR04.pdf>

Figura 15 – Foto da placa de interface utilizada



Fonte: (RIBEIRO, 2017)

Figura 16 – Placa MD-25



Fonte: <https://www.robot-electronics.co.uk/htm/md25tech.htm>

### 3.2 Parte computacional

A parte computacional do robô diz respeito a todo *software* usado para a programação do controlador implementado no sistema embarcado no robô usado neste trabalho, que neste caso se trata do microcontrolador ATmega328P, contido na placa Arduino Uno. As especificações deste microcontrolador são as seguintes<sup>6</sup>:

- 32 pinos para uso com finalidades de propósito geral e alimentação (sendo 6 pinos capazes de receber entradas analógicas);
- Memória de programa flash de 32 Kb;
- Velocidade da CPU (*Central Processing Unit*) de 20 MIPS (*Millions of Instructions Per Second*);
- Memória RAM (*Random Access Memory*) de 2 Kb;
- Memória EEPROM (*Electrically Erasable Programmable Read-Only Memory*) de 1 Kb;
- Periféricos de comunicação: 1 canal I2C, 2 canais SPI (*Serial Peripheral Interface*) e 1 canal UART;
- 2 timers de 8 bits, 1 timer de 16 bits;
- 6 canais PWM (*Pulse Width Modulation*);
- Baixo consumo de energia (em comparação com o ATmega328).

A programação do ATmega328P pode ser feita pela IDE disponibilizada no *site* do Arduino<sup>7</sup>. Nesta IDE, escolhe-se qual Arduino se deseja programar. No caso deste trabalho, escolheu-se a opção “Arduino/Genuino Uno” (Figura 17). Os códigos são feitos em C/C++, com algumas funções e bibliotecas próprias da interface.

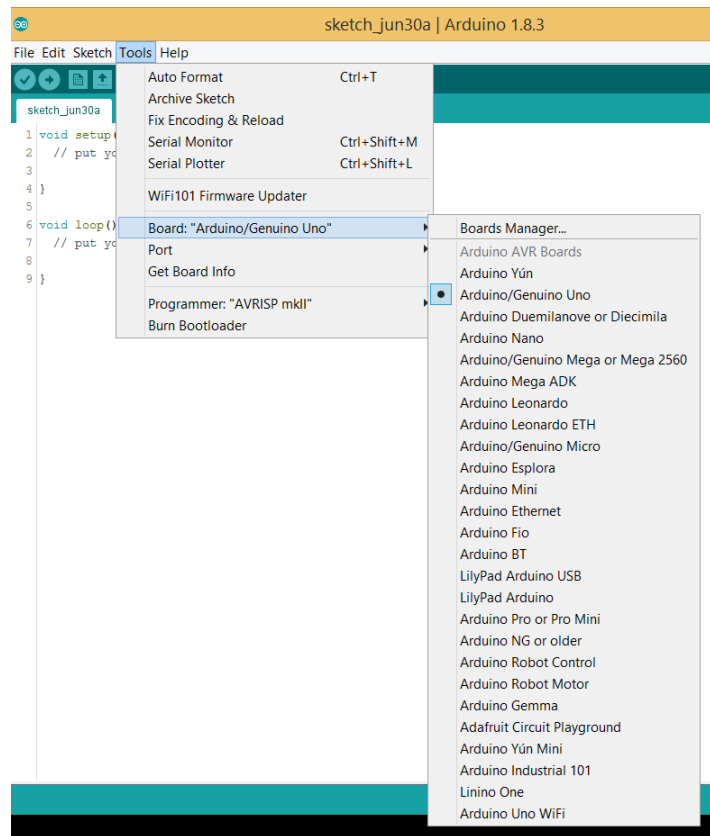
A principal biblioteca usada na programação do sistema móvel foi a “**Wire.h**”, responsável pela comunicação I2C do Arduino com outros periféricos que também utilizam este protocolo (no caso, a placa MD-25). Para a comunicação *Bluetooth*, não foi necessária a inclusão de novas bibliotecas em código, pois se utilizou das funções da biblioteca “Serial”, automaticamente inclusas pela própria IDE.

---

<sup>6</sup> Site: <http://www.microchip.com/wwwproducts/en/ATmega328p>

<sup>7</sup> Site: <https://www.arduino.cc/>

Figura 17 – Seleção da placa utilizada na IDE.

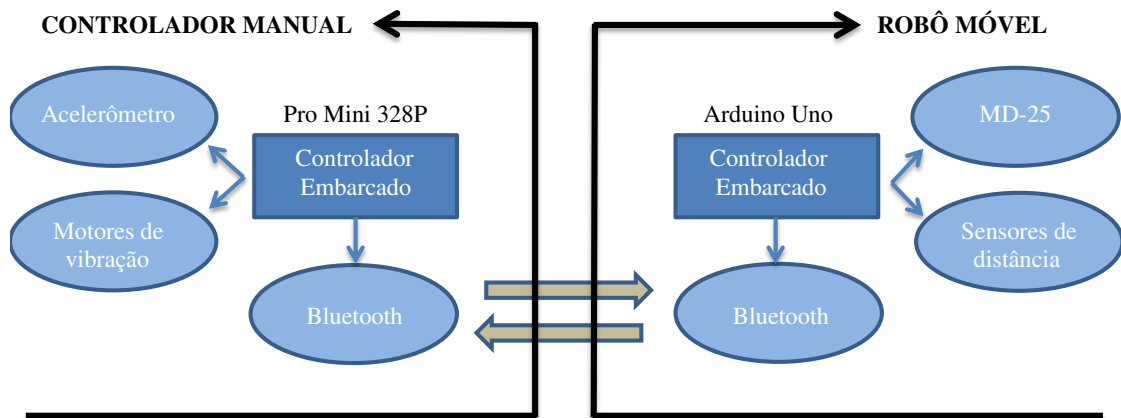


Fonte: AUTOR

A Figura 18 apresenta um diagrama contendo os elementos de ambos os sistemas em conjunto. Neste diagrama, percebe-se que na parte do robô móvel se usa o Arduino Uno, enquanto que no controlador manual com *feedback* táctil se usa a placa Arduino Pro-Mini 328P para o processamento do seu módulo de *software* (detalhado no Capítulo 4). Os motivos para esta escolha serão explicados no capítulo seguinte. Na Figura 19, é apresentado um diagrama de blocos exemplificando o funcionamento do código de controle do robô. O código em C/C++ comentado em detalhes pode ser encontrado no Apêndice A deste trabalho.

No início do código encontram-se inclusão de bibliotecas, declaração e inicialização de variáveis, protótipos de funções e definição de pinos como entrada ou saída. Já dentro do laço infinito do código, a primeira etapa é receber os dados do acelerômetro (eixos X e Y), via *Bluetooth*, provenientes do controlador manual. Esses dados são utilizados para decidir em qual direção o robô se moverá. A Tabela 1 mostra quais movimentos são feitos pelo robô, relacionados com as respectivas leituras do acelerômetro (os intervalos dos valores das variáveis de leitura do acelerômetro utilizadas na tabela – AcX e AcY – podem ser

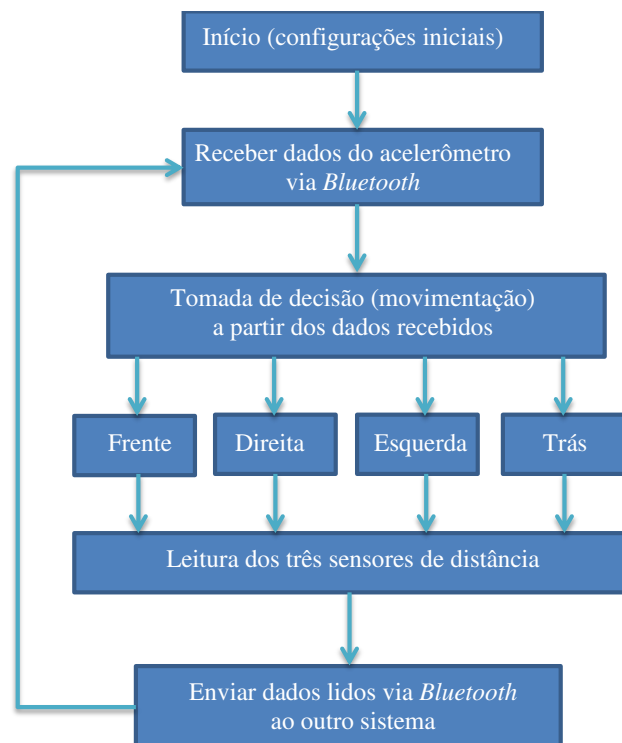
Figura 18 – Diagrama de blocos dos dois sistemas em conjunto.



Fonte: AUTOR

consultados no código disponível no Apêndice A.6). A última etapa consiste em ler os três sensores de distância (infravermelho na parte da frente do robô e dois sonares, um em cada uma de suas laterais). Os valores das distâncias lidos e enviados para o controlador manual são utilizados na ativação dos respectivos motores de vibração, realizando um *feedback* tátil.

Figura 19 – Diagrama de blocos do código de controle do robô móvel.



Fonte: AUTOR



Tabela 1 – Movimentos do robô relacionados aos valores codificados do acelerômetro

		Codificações AcX (Eixo X)		
		-----	Baixo	Centralizado
Codificações AcY (Eixo Y)	Esquerda	Frente-esquerda	Esquerda	Ré-esquerda
	Centralizado	Frente	Parado	Ré
	Direita	Frente-direita	Direita	Ré-direita

Fonte: AUTOR

Na codificação, o motor vibrará mais forte quanto menor for a distância medida pelo respectivo sensor.

No próximo capítulo, serão detalhados os elementos do controlador manual com *feedback* tátil, assim como a sua programação, que será detalhada em diagrama de blocos, abordando com maior aprofundamento a comunicação entre os sistemas.

## 4 IMPLEMENTAÇÃO DO SISTEMA DE COMANDO DO ROBÔ NA LUVÁ

O sistema de comando projetado tem a finalidade de controlar os movimentos do robô móvel através de gestos da mão de um usuário, além de receber *feedback* tátil enviado por sensores de distâncias no robô, informando-o se existem objetos nas proximidades do mesmo. Tendo em vista o objetivo deste sistema, definiram-se os componentes que foram utilizados, além do projeto e confecção do *design* para este sistema. Uma luva foi utilizada para acomodar o sistema e os elementos de *feedback* tátil em uma das mãos da pessoa que controla os movimentos do robô. Por último, programou-se o microcontrolador utilizado para que realizasse as funções previstas.

### 4.1 Componentes utilizados

Para o sistema de controle acoplado na luva foram definidos os seguintes componentes:

- a) Placa Pro Mini 328P (microcontrolador ATmega328P);
- b) Motor de vibração 1027 (responsável pelo *feedback* tátil);
- c) Módulo acelerômetro/giroscópio MPU-6050;
- d) Módulo *Bluetooth* HC-05;

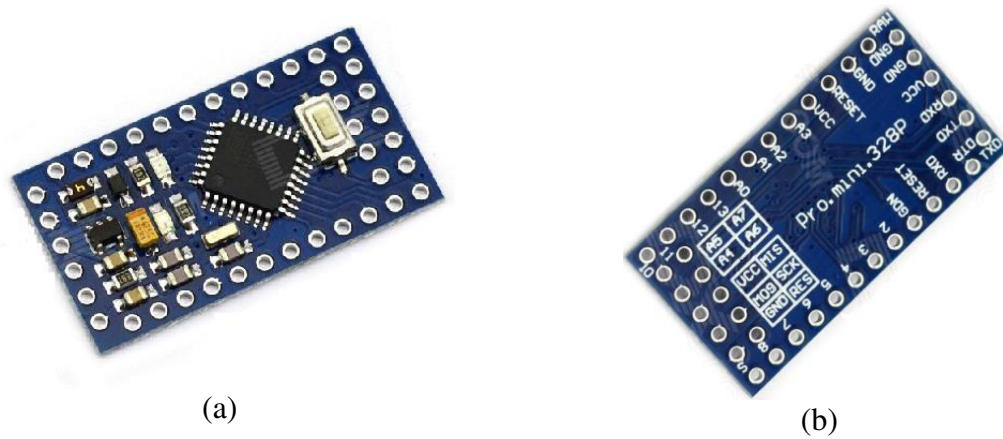
O microcontrolador utilizado no controlador manual é o mesmo usado no robô móvel, porém neste caso será utilizado na placa Pro Mini 328P (Figura 20), pois esta tem tamanho reduzido em comparação à primeira, mantendo o *design* do sistema menor. É importante ressaltar que, por se tratar de outra placa, é necessário especificá-la na IDE de programação, assim como foi feito para a placa Arduino Uno anteriormente. Os motivos de escolha deste dispositivo são os mesmos já explicitados no Capítulo 3, onde também se encontram suas especificações técnicas.

O motor 1027 é um pequeno dispositivo atuador utilizado em aplicações de vibração (como em celulares), que opera a uma tensão de 3,3 V com corrente média de 90 mA<sup>8</sup>. Existem outros motores com a mesma finalidade, porém o tamanho dos mesmos é relativamente grande, inviabilizando sua utilização na aplicação proposta que prevê o contato do motor com a mão humana provendo o *feedback* tátil. Pode-se visualizar o motor, além de seu tamanho real, na Figura 21.

---

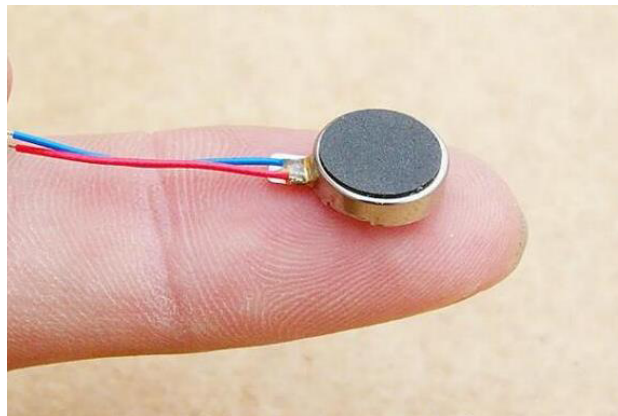
<sup>8</sup> Site: <http://www.filipeflop.com/pd-361d89-motor-de-vibracao-1027.html?ct=&p=1&s=1>

Figura 20 – Placa Pro Mini 328p a) frente e b) costa



Fonte: <http://www.dx.com/p/jtron-arduino-pro-mini-atmega328p-5v-16m-electronic-building-block-module-blue-375931#>

Figura 21 – Motor de vibração 1027



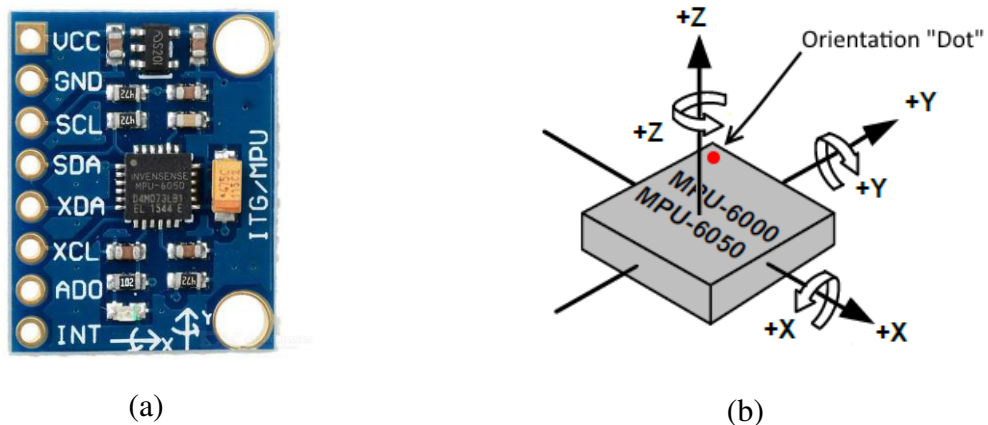
Fonte: <https://pt.aliexpress.com/item/10-2-7MM-Button-type-vibrating-motor-1027/32613965351.html?spm=2114.42010208.4.58.Guq0EJ>

Para se controlar o robô móvel através de gestos da mão, é necessário utilizar um acelerômetro, que consegue quantificar a aceleração linear em diferentes eixos e enviá-la ao chip MPU-6050<sup>9</sup> (*MotionTracking device*). Esse módulo possui função de acelerômetro (três eixos), giroscópio (três eixos) e processamento digital de movimento (estas duas últimas não foram utilizadas neste trabalho). Por meio do protocolo I2C, consegue-se acessar os registradores do módulo e adquirir as leituras desejadas, que, no caso deste trabalho, são apenas os valores das acelerações nos eixos X e Y (robô move-se no plano). Esses valores serão alterados conforme a movimentação do módulo, através dos gestos do usuário, resultados da decomposição nos eixos do valor da aceleração da gravidade. Assim, foi

<sup>9</sup> Site: <https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>

estabelecida uma relação entre os valores lidos e os movimentos do robô. Na Figura 22 é mostrado o módulo MPU-6050, assim como uma ilustração de seus eixos (X, Y e Z) de onde se obtém os valores do acelerômetro e do giroscópio.

Figura 22 – a) Módulo MPU-6050 b) Eixos do módulo



Fonte: <https://www.ardunity.com/single-post/2016/11/17/42-MPU-60509250>

Por último, tem-se o módulo *Bluetooth* HC-05, o mesmo que é utilizado no robô móvel, já detalhado no Capítulo 3.

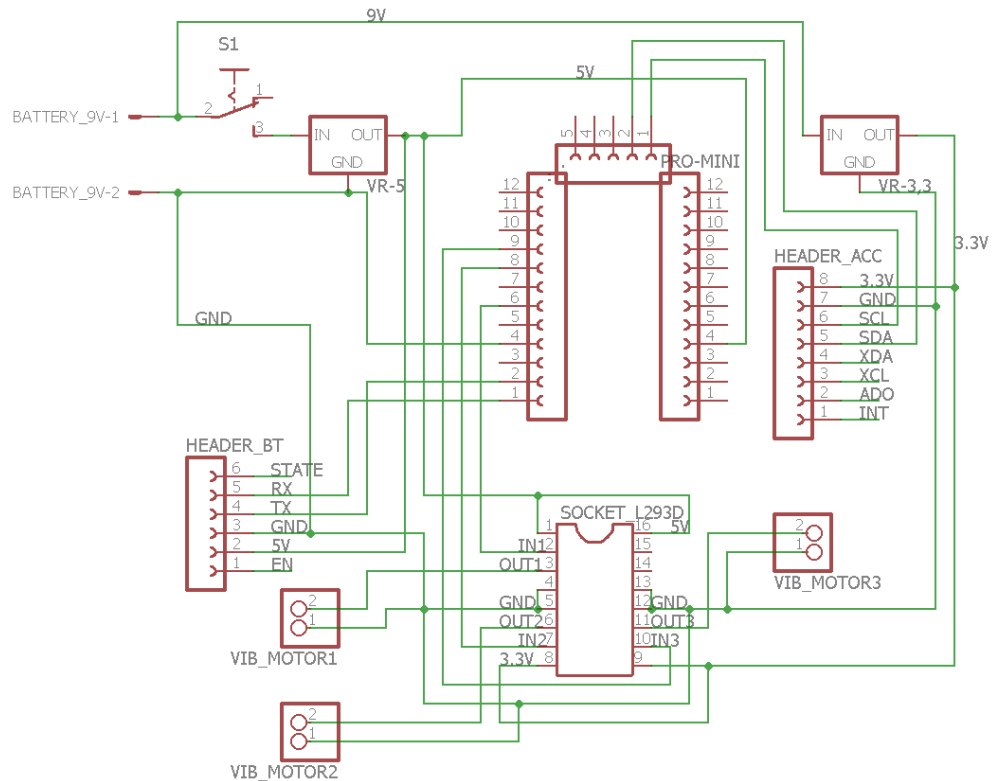
## 4.2 Placa de circuito do sistema de comando

Antes de se construir a placa de circuito do sistema do controlador manual, deve-se seguir alguns passos importantes, como o projeto e a prototipação do circuito elétrico. Logo, o primeiro passo foi projetar o sistema em um *software* de projeto de circuitos, que fez uso do programa Eagle<sup>10</sup>. O projeto final pode ser visto na Figura 23.

Além dos elementos descritos no item anterior, o circuito também conta com um regulador de tensão de 3,3 V (LF33ABV), necessário para alimentação dos motores de vibração e do MPU-6050, e um regulador de tensão de 5 V (7805), utilizado para alimentação do módulo *Bluetooth* HC-05 e da placa Pro Mini 328P. A alimentação principal será feita por uma bateria recarregável de níquel-hidreto metálico de 9 V. Também foi inserido no circuito um push-button do tipo retenção, conectado entre o terminal positivo da bateria e o ponto de alimentação de 9 V do circuito (que são as entradas dos reguladores de tensão), tornando mais prático o processo de ligar e desligar do sistema.

<sup>10</sup> Site: <http://www.eletrica.ufpr.br/mehl/pci/apostila2cc.pdf>

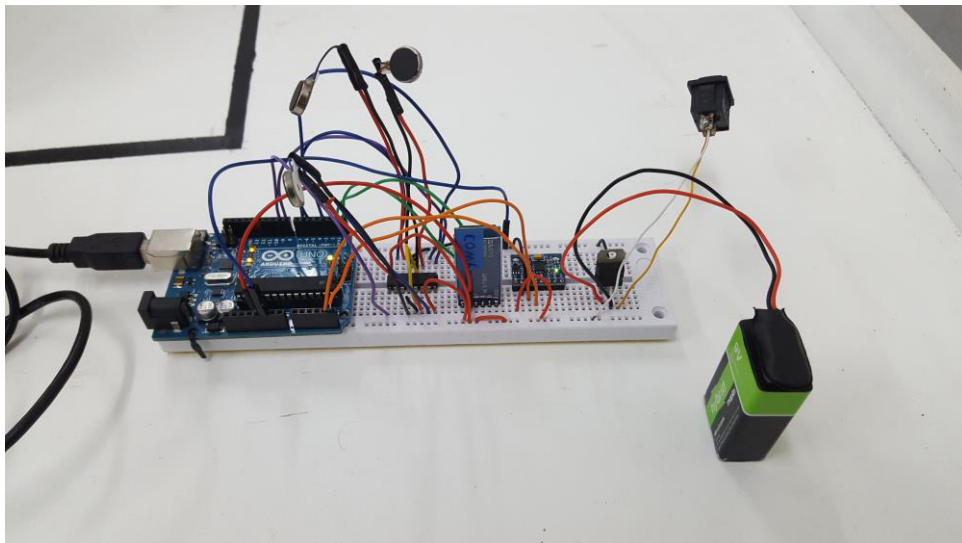
Figura 23 – Projeto de circuito do sistema do controlador manual



Fonte: AUTOR

O próximo passo é a montagem do circuito projetado na forma de protótipo (Figura 24), o que geralmente é feito utilizando-se uma *protoboard*. Na prototipagem foi usado um

Figura 24 – Protótipo de circuito para o sistema de controlador manual com *feedback* tátil

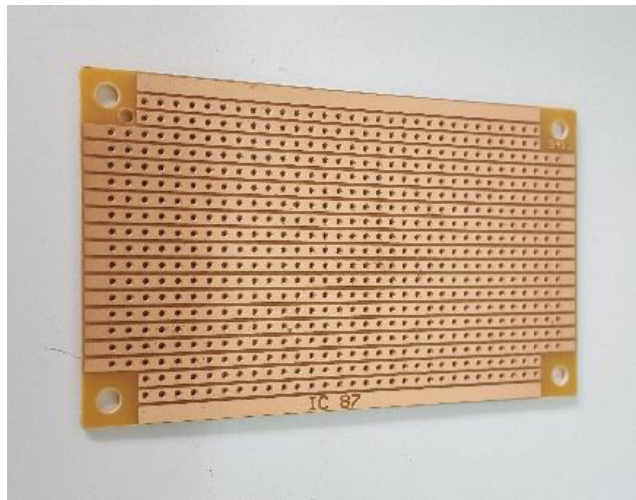


Fonte: AUTOR

Arduino Uno no lugar da placa Pro Mini 328P. Vários testes com protótipos incompletos foram feitos até o teste final com o protótipo completo, como será visto no Capítulo 5.

Por último, para a construção da placa de circuito, soldaram-se todos os componentes em uma placa universal (Figura 25), removendo assim a utilização da *protoboard* e reduzindo o tamanho do sistema. A imagem seguinte, presente na Figura 26, mostra a placa de circuito pronta, depois de feitas todas as soldagens.

Figura 25 – Placa universal utilizada



Fonte: AUTOR

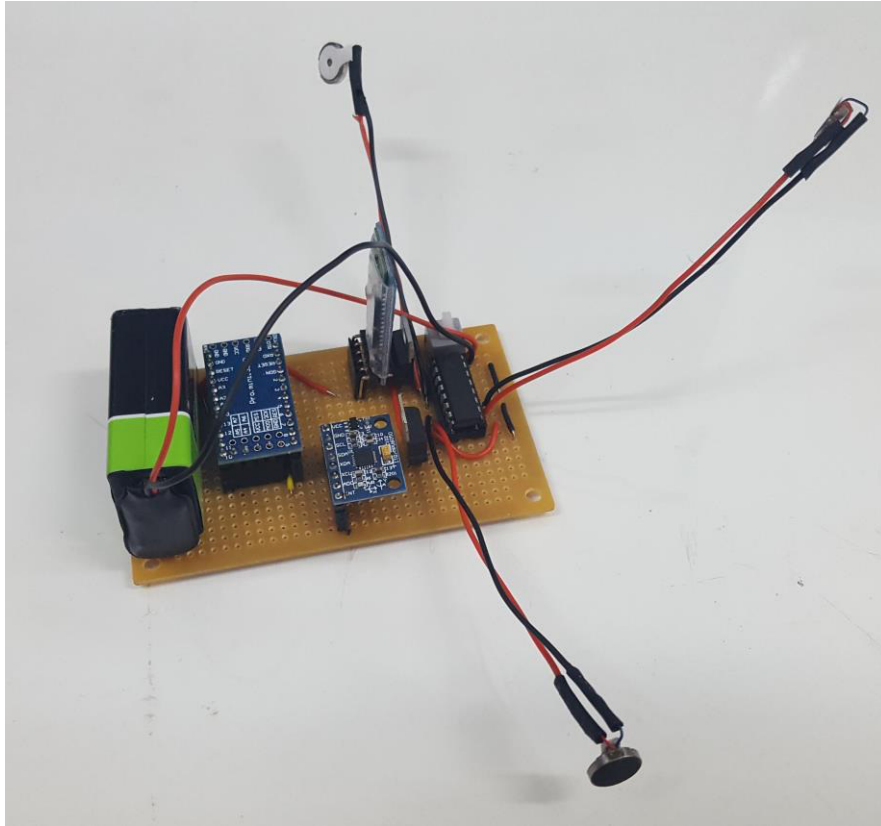
### 4.3 Acoplamento do sistema de comando à luva

Após a montagem da placa de circuito, foi feito o seu acoplamento de forma fixa em uma luva, com o propósito de implementar o *feedback* táctil. Nessa montagem, os motores de vibração foram dispostos na parte interna da luva, de forma a ficar em contato direto com a mão do usuário. Assim, quando os motores forem ativados, a vibração será mais claramente sentida pelo operador. A Figura 27a mostra o sistema do controlador manual completo (placa de circuito fixada à luva), e a Figura 27b mostra o mesmo sendo utilizado por um operador.

### 4.4 Parte computacional

Para que o sistema realize as funções propostas, deve-se programar o microcontrolador do circuito para executar as funções desejadas. A Figura 28 mostra o diagrama da

Figura 26 – Placa de circuito do controlador manual



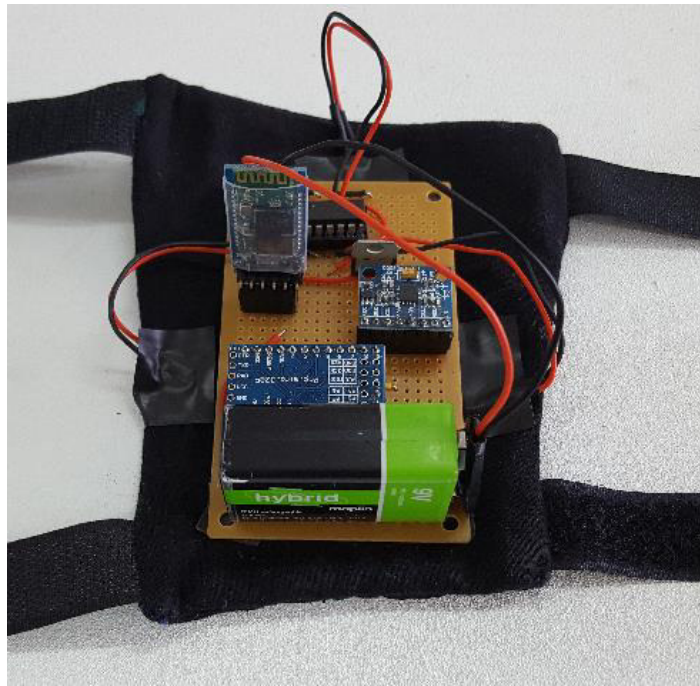
Fonte: AUTOR

programação feita para o controlador manual, que é similar ao diagrama mostrado para o robô móvel no Capítulo 3.

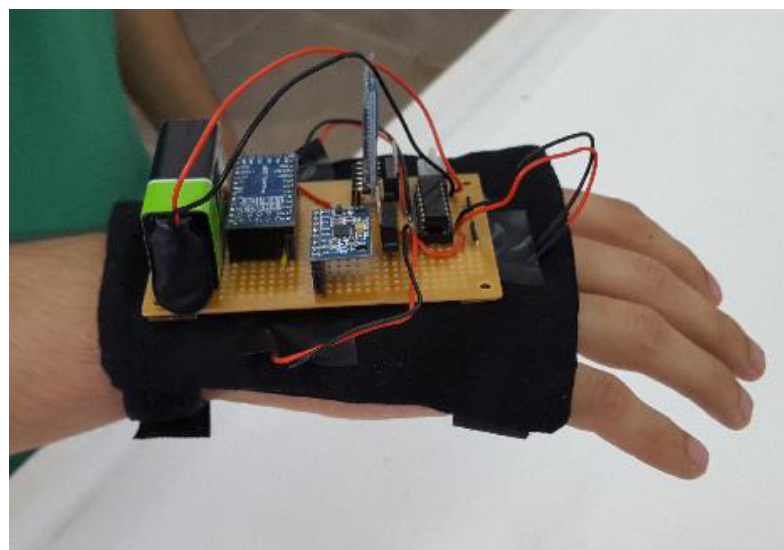
O início do programa contém inclusão de bibliotecas, declaração e inicialização de variáveis, protótipos de funções e definição de pinos como entrada ou saída, exatamente como o sistema do robô móvel. A primeira etapa dentro do laço infinito do programa é a leitura dos valores do acelerômetro (eixos X e Y). Em seguida estes dados são codificados e enviados, via *Bluetooth*, para o robô móvel. A segunda etapa é receber os dados dos sensores provenientes do robô, para assim utilizá-los na tomada de decisão de quais motores de vibração serão ativados. Conforme mencionado no Capítulo 3, os valores enviados ao robô de forma codificada permitem a realização de nove movimentos em direções distintas, porém em velocidades constantes. Similarmente, a intensidade da vibração de cada motor depende da mensagem codificada de distância lida pelos sensores sonares e infravermelho. A Tabela 2 seguinte mostra exatamente isso: para distâncias maiores que 25 cm, os motores não vibram; para distâncias entre 15 cm e 25 cm, os motores vibram com baixa intensidade; e para distâncias menores que 15 cm, os motores vibram com maior intensidade. Os valores de

distância de 15 cm e 25 cm foram escolhidos de forma arbitrária, porém considerando um possível atraso na resposta do usuário ao sentir a vibração. Estes valores de intensidade fraca ou forte serão explicados no Capítulo 5, na seção que trata sobre os testes dos motores de vibração.

Figura 27 – a) Sistema do controlador manual com *feedback* táctil b) sistema sendo utilizado por um operador



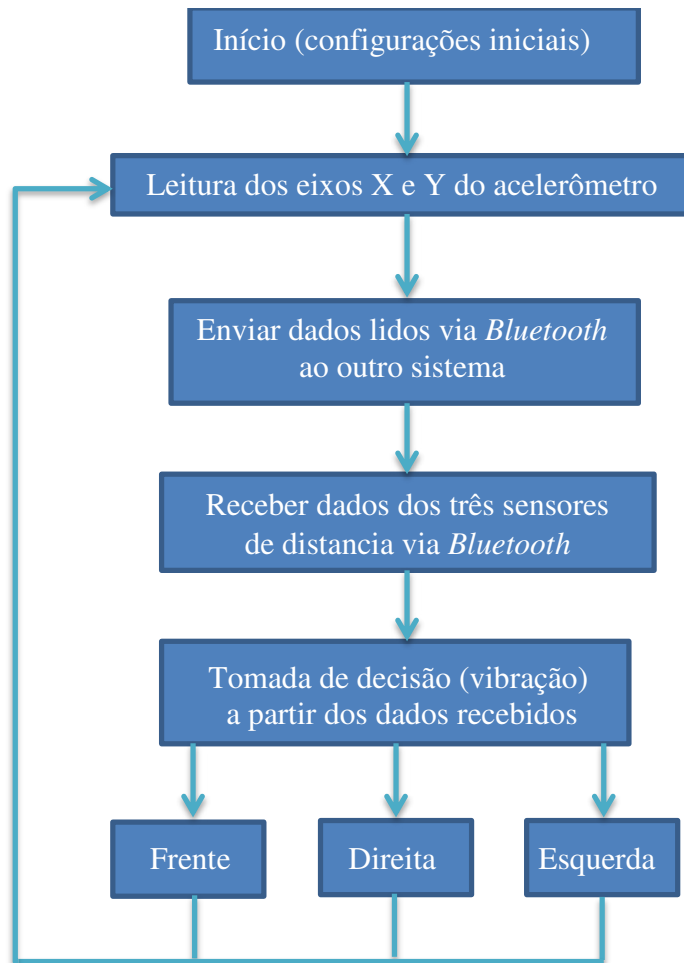
(a)



(b)



Figura 28 – Diagrama de blocos da programação do controlador manual



Fonte: AUTOR

Tabela 2 – Intensidade da vibração dos motores relacionada aos valores codificados das distâncias lidas

		Codificações de distâncias D (em cm)			
		-----	<b>D &gt; 25</b>	<b>15 &lt; D &lt; 25</b>	<b>D &lt; 15</b>
Motores de vibração	<b>Esquerda</b>		Desligado	Vibração fraca	Vibração forte
	<b>Cima</b>		Desligado	Vibração fraca	Vibração forte
	<b>Direita</b>		Desligado	Vibração fraca	Vibração forte

Fonte: AUTOR

No caso do robô móvel, primeiro são esperados os dados do outro sistema (movimentos codificados) para depois enviar os dados dos sensores. Já no controlador manual é feito primeiro o envio dos dados do acelerômetro, para posteriormente receber pelos dados dos sensores de distâncias. Isto é feito para que os dois sistemas não fiquem travados, um esperando por dados do outro, sendo o controlador manual o responsável por iniciar o ciclo entre os sistemas.

## 5 EXPERIMENTOS E RESULTADOS

Vários experimentos foram conduzidos a fim de testar o funcionamento tanto dos componentes utilizados nos sistemas quanto dos sistemas (controle manual e robô) operando em conjunto. Os primeiros testes foram feitos montando protótipos em *proto-board*, enquanto os testes finais foram conduzidos com os sistemas finalizados. Ao total foram realizados nove experimentos: Comunicação entre dispositivos *Bluetooth*, acelerômetro, sensores de distância, envio de dados do acelerômetro via *Bluetooth*, acionamento de motores de vibração, acionamento de motores do robô móvel, controle de movimento do robô, *feedback* táctil e sistema completo.

Além dos testes para a verificação do funcionamento, foram feitos 20 testes de repetibilidade em todos os experimentos, incluindo o do sistema completo. O objetivo da repetibilidade das ações feitas nos testes deu-se em virtude de estatisticamente comprovar a robustez desses sistemas em suas atribuições, não só de forma isolada quanto integrada. Além disso, buscou-se verificar, relatar e justificar possíveis erros aleatórios que pudessem ser observados em todos os testes.

### 5.1 Comunicação entre dispositivos *Bluetooth*

Este experimento consiste no estabelecimento da comunicação entre os dispositivos *Bluetooth*, de forma que um módulo consiga enviar dados para o outro. A partir de um exemplo de configuração dos módulos<sup>11</sup>, fixou-se um dos módulos como mestre e outro como escravo, além de ter sido feito um *bind* entre os módulos, ou seja, cada módulo conecta-se apenas com outro, especificado por um endereço fixo inerente a cada dispositivo. Os códigos de teste dos módulos *Bluetooth* do transmissor e receptor encontram-se no Apêndice A.1 e A.2, respectivamente.

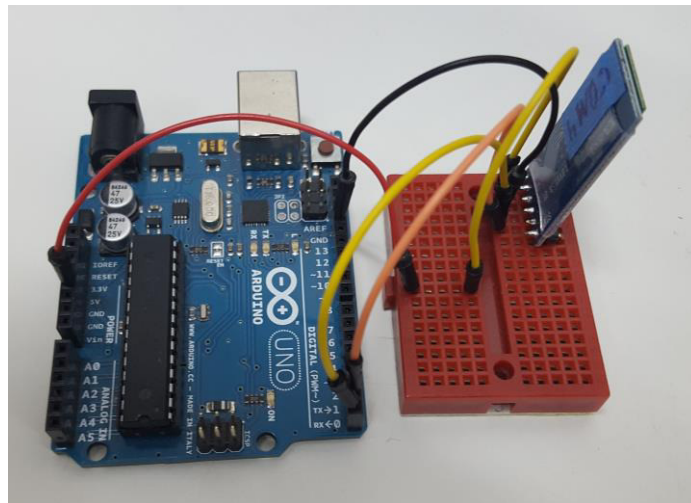
Ambos os módulos *Bluetooth* foram conectados cada um a um Arduino Uno, de forma que o pino 'Tx' do módulo é conectado ao pino 'Rx' do Arduino (Figura 29). Os códigos utilizados consistem no simples envio de um valor inteiro qualquer, por parte do Arduino transmissor, e no recebimento e impressão deste valor em tela, por parte do Arduino receptor.

Para os testes de repetibilidade desta comunicação foram feitas pequenas alterações nos códigos originais (Apêndices A.1 e A.2), de forma a estabelecer uma comunicação bilateral

---

<sup>11</sup> Site: <http://www.instructables.com/id/How-to-Configure-HC-05-Bluetooth-Module-As-Master-/>

Figura 29 – Ligação entre o Arduino Uno e o módulo HC-05



Fonte: AUTOR

entre os módulos, ou seja, cada módulo envia e recebe informações do outro, de forma que um só envia dados após receber dados do outro. Em cada teste manteve-se a comunicação estabelecida durante um minuto. Na totalidade dos 20 ensaios de comunicação não houve perdas de dados, pois a transferência se deu de forma contínua em ambos os sentidos. Caso fossem perdidos dados, ambos os módulos “travariam”, um esperando por dados do outro.

## 5.2 Acelerômetro

Os experimentos realizados com o módulo MPU-6050 têm como objetivo somente a leitura do acelerômetro, recebendo os dados relacionados aos eixos X e Y. Conforme comentado no Capítulo 4, esse dispositivo possui outras funções não utilizadas neste trabalho. O código de teste para leitura do acelerômetro encontra-se no Apêndice A.3. A leitura deste dispositivo é feita via barramento I2C, cujas funções no Arduino são declaradas na biblioteca ‘**Wire.h**’ e definidas no código ‘**Wire.cpp**’, ambos desenvolvidos para operar no Arduino.

O teste de repetibilidade foi feito executando 500 leituras por teste e de cada teste foram extraídos o valor máximo e mínimo lido de cada eixo. Foram feitos testes para o módulo em três diferentes inclinações ( $0^\circ$ ,  $45^\circ$  e  $90^\circ$ ). Após a coleta dos dados, calcularam-se a média e o desvio padrão dos ensaios, para as diferentes posições e para os dois eixos. A Tabela 3 mostra os resultados obtidos para o eixo X (os resultados para o eixo Y são similares). Os valores brutos lidos pelo acelerômetro variam de -32768 a 32767 (16 bits de resolução) e correspondem a uma porcentagem da sensibilidade configurada (a configuração padrão é de

Tabela 3 – Análise dos dados coletados do acelerômetro para o eixo X

Inclinação do módulo			
-----	0°	45°	90°
<b>Valor médio</b>	1812	13027	17113
<b>Desvio padrão</b>	232	301	281
<b>Desvio padrão (%)</b>	13,55%	2,31%	1,64%

Fonte: AUTOR

2g – duas vezes a aceleração da gravidade), ou seja, uma leitura de 16383 corresponde a uma aceleração igual à da gravidade. Percebe-se, por meio dos valores médios, que a relação entre o valor lido do acelerômetro para o eixo X e a inclinação do módulo é uma relação não-linear. O mesmo procedimento foi feito para o eixo Y obtendo resultados similares.

### 5.3 Sensores de distância

Os sensores de distância (infravermelho e sonar) foram testados a fim de adquirir suas leituras de distância. Para isto, utilizou-se o código presente no Apêndice A.4 para o sensor ultrassônico (sonar) e o código do Apêndice A.5 para o sensor infravermelho. Ainda que esses dados não têm implicações diretas no sistema proposto (controle por *feedback* tátil), os testes foram conduzidos como forma de validá-los para a mensuração de distância, sendo ambos considerados de baixo custo.

Os testes de repetibilidade foram feitos verificando se as leituras dos sensores eram consistentes e não variavam significativamente para um objeto fixo posicionado em três diferentes posições (15, 30 e 45 cm). Cada teste foi feito em um intervalo de 10 segundos, sendo feitas 10 medidas por segundo, totalizando 100 medições por teste. As Tabelas 4, 5 e 6 mostram os erros relativos a cada distância ao objeto lida pelos sensores, respectivamente o infravermelho, sonar direito e sonar esquerdo. Percebe-se que houve um erro médio baixo no pior caso, que foi de 3,56% (46,6/45) para o sonar esquerdo. Os desvios padrões para todos os sensores nas distâncias lidas também apresentaram erros baixos, ainda que percentualmente (relativo às distâncias medidas) tenham alcançado 11,76% no pior caso (distância de 15 cm para o sonar direito).

Tabela 4 – Análise de medições do infravermelho

Distância do objeto			
-----	<b>15 cm</b>	<b>30 cm</b>	<b>45 cm</b>
<b>Valor médio</b>	15,2 cm	30,6 cm	45,6 cm
<b>Desvio padrão</b>	0,8 cm	1 cm	1,4 cm
<b>Desvio padrão (%)</b>	5,26%	3,27%	3,07%

Fonte: AUTOR

Tabela 5 – Análise de medições do sonar direito

Distância do objeto			
-----	<b>15 cm</b>	<b>30 cm</b>	<b>45 cm</b>
<b>Valor médio</b>	15,3 cm	30,9 cm	46,3 cm
<b>Desvio padrão</b>	1,8 cm	2,5 cm	3,0 cm
<b>Desvio padrão (%)</b>	11,76%	8,09%	6,48%

Fonte: AUTOR

Tabela 6 – Análise de medições do sonar esquerdo

Distância do objeto			
-----	<b>15 cm</b>	<b>30 cm</b>	<b>45 cm</b>
<b>Valor médio</b>	15,4 cm	30,8 cm	46,6 cm
<b>Desvio padrão</b>	1,7 cm	2,4 cm	3,2 cm
<b>Desvio padrão (%)</b>	11,04%	7,79%	6,87%

Fonte: AUTOR

#### 5.4 Envio de dados do acelerômetro via *Bluetooth*

Neste experimento já estão integrados dois dos testes anteriores: comunicação *Bluetooth* e leitura do acelerômetro. O objetivo deste tipo de experimento é verificar o recebimento dos dados de controle do movimento do robô.

Os valores da leitura do acelerômetro foram codificados em números de um algarismo inteiro (16 bits) para cada eixo (X e Y) antes de serem enviados, tornando mais simples o controle realizado pelo dispositivo. Conforme apresentado na Tabela 1 do Capítulo 3, com esses valores foi possível definir nove movimentos para comandar o robô. Os valores da codificação são definidos em função das faixas dos dados lidos do acelerômetro: menores que -6000, entre -6000 e 6000, e maiores que 6000, definindo três posições para cada eixo. Os códigos deste experimento (transmissor e receptor) encontram-se no Apêndice A.6 e A.7.

Para o teste de repetibilidade, manteve-se o MPU-6050 sempre estático, de forma que os valores codificados da leitura seriam sempre os mesmos. Além disto, alterou-se o código do transmissor para que houvesse a comunicação bilateral (similarmente ao teste da Seção 5.1), de forma que um módulo espera por dados do outro para continuar o processamento do código. Os dados codificados do eixo X e Y foram enviados 5000 vezes em cada um dos 20 testes, e um contador foi adicionado ao código do receptor, de forma que quando a leitura fosse diferente da esperada, o contador seria incrementado. Assim como no teste da Seção 5.1, não houve problemas na comunicação, pois na totalidade dos dados enviados, eles foram recebidos corretamente (o contador de erros foi igual a zero).

#### 5.5 Acionamento de motores de vibração

Os motores de vibração foram acionados utilizando-se o circuito integrado de drive de motores L293D, sendo possível comandar tanto a velocidade quanto a direção de motores. Nesta aplicação que envolve a vibração do motor, não faz sentido o controle de direção que é realizado em relação ao eixo de motores (horário ou anti-horário). Logo, foi controlada apenas sua velocidade de vibração realizada através da tensão DC média aplicada a seus terminais. O código para teste deste componente encontra-se no Apêndice A.8. Os pinos de *Enable* do L293D foram utilizados sempre em estado alto, enquanto os pinos de entrada (referenciados como IN1, IN2 e IN3 no esquemático da Figura 23 no Capítulo 4) receberam o sinal PWM que controla a velocidade dos motores, através da tensão média desse sinal.

Nos testes, a vibração do motor que está relacionada à informação de *feedback* tátil, pode ser aferida indiretamente pela tensão lida nos terminais do motor. Quanto maior for, maior será a vibração e, neste caso, menor a distância do referido sensor do robô (frente ou das laterais) ao obstáculo. Seu objetivo, portanto, é validar os dados de vibração, de forma indireta, para as distâncias programadas para os três sensores.

A Tabela 7 a seguir mostra os resultados de medição de tensão média nos motores de vibração, relacionados ao respectivo *Duty Cycle* utilizado no sinal de PWM de controle dos mesmos motores. A partir dos dados obtidos, pode-se concluir que a relação *Duty Cycle* e tensão média é uma linear.

Tabela 7 – Relação entre tensão média nos motores e *Duty Cycle* utilizado

		Valores de <i>Duty Cycle</i> utilizados (0 a 255)				
		-----	<b>100</b>	<b>140</b>	<b>180</b>	<b>220</b>
Valores de tensão média (V)	<b>Motor 1</b>	1,10	1,45	1,88	2,25	2,60
	<b>Motor 2</b>	1,08	1,50	1,90	2,28	2,62
	<b>Motor 3</b>	1,08	1,49	1,90	2,27	2,62

Fonte: AUTOR

## 5.6 Acionamento de motores do robô móvel

Este teste foi feito para verificar a movimentação do robô móvel RoSeLi, que é feita a partir de comandos I2C enviados à placa MD-25. O teste consistiu em simplesmente movê-lo para frente, para trás e girar para ambos os lados, além de parar por uma pequena duração de tempo entre um movimento e outro. O código implementado encontra-se no Apêndice A.9. É importante ressaltar que cada roda do robô é acionada individualmente (acionamento diferencial) com valores distintos. No acionamento individual é possível comandar movimentos isolados (por exemplo, frente) ou combinados (frente e direita). Logo, para os casos de movimentos curvos basta reduzir a velocidade de uma das rodas em relação à outra.

Os testes de repetibilidade mostraram o correto funcionamento desta movimentação, tendo o robô movimentado conforme programado.



## 5.7 Controle de movimento do robô

Este teste já faz parte do conjunto de experimentos finais, onde se valida não só a operação dos componentes individuais, envolvendo mais de um sistema. Neste teste, foi feito o controle dos movimentos do robô RoSeLi utilizando os dados codificados da leitura do acelerômetro, recebidos via *Bluetooth*. O código para este experimento encontra-se no Apêndice A.10, enquanto o do envio dos dados do acelerômetro no Apêndice A.6.

Em todos os testes de repetibilidade feitos, os movimentos obedeceram aos comandos enviados pelo acelerômetro, tendo a comunicação funcionada corretamente e sem falhas.

## 5.8 *Feedback* táctil

O sistema de *feedback* táctil consiste no recebimento da leitura dos sensores de distância (infravermelho e sonares), vindos do robô móvel, em conjunto com a atuação dos motores de vibração, de acordo com os valores recebidos. Além disto, este sistema também induz o usuário a fazer uma possível alteração no controle de movimento do robô, a depender da realimentação recebida pelos motores de vibração. Por exemplo, caso o motor vibre informando obstáculo próximo à frente do robô, o usuário poderia mover a mão comandando movimento de ré ou para um dos lados de maneira a evitar colidir com o obstáculo.

O *feedback* táctil foi definido da seguinte maneira: caso a distância lida pelo sensor seja maior que 25 cm, a vibração não será acionada. Caso esta distância esteja entre 15 cm e 25 cm, é acionado o motor de vibração com um PWM de baixo *duty cycle*. No caso de distâncias menores que 15 cm é acionado o motor com um PWM de alto *duty cycle*.

O código para este sistema encontra-se no Apêndice A.11, sendo o programa de envio dos dados dos sensores o mesmo do sistema do robô móvel, o qual será comentado na Seção 5.9.

Nos testes de repetibilidade realizados, os motores de vibração funcionaram de acordo com os dados enviados pelos sensores na totalidade dos casos. Não houve problemas quanto ao sinal PWM enviados aos motores ou quanto ao acionamento dos mesmos.

## 5.9 Sistema completo

Denomina-se por sistema completo a interação entre ambos os sistemas desenvolvidos neste trabalho: controlador manual e robô móvel redefinido (sistema embarcado e placa de interface, conforme mencionado no Capítulo 3). Este experimento é o último e definitivo, onde se alcança o objetivo desejado, no caso, *feedback* tátil e controle do robô por gestos manuais.

O sistema do controlador manual, detalhado no Capítulo 4 e na Seção 5.8, é composto pelo envio de dados do acelerômetro, recebimento dos dados dos sensores de distância e utilização destas informações para as ativações dos três motores de vibração (*feedback* tátil) presentes no sistema.

O código final para o sistema do robô móvel consiste numa pequena alteração do código presente no Apêndice A.10 (controle de movimento do robô), tendo sido adicionado a leitura dos sensores (dois sonares e um infravermelho) e o envio destes dados, via *Bluetooth*, para o outro sistema (controlador manual). Este código pode ser encontrado no Apêndice A.12.

Os testes de repetibilidade foram realizados com o intuito de conferir a robustez e confiabilidade do sistema, verificando possíveis problemas na comunicação, no acionamento dos dispositivos ou nos sinais enviados entre os sistemas. Após a realização desses testes, não se percebeu qualquer um dos problemas citados acima. O robô obedeceu aos comandos do acelerômetro, e a vibração foi realizada corretamente, tanto em intensidade quanto em escolha do motor a ser acionado.

## 6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho de monografia foi proposto e desenvolvido um sistema de controle de movimento de um pequeno robô móvel por gestos manuais, contendo *feedback* tátil relacionado a objetos próximos ao robô, sendo a comunicação entre os programas localizados na mão do operador e embarcado no robô feita através do sistema *Bluetooth*. Na implementação da proposta mecatrônica, foi empregado o robô RoSeLi disponível no LRC, onde essa pesquisa foi desenvolvida. O sistema embarcado no robô e a placa de interface foram modificados em relação à original existente, sendo a placa Arduino empregada na execução do programa embarcado no robô. A placa de interface existente também teve que ser adaptada aos propósitos do projeto. O controlador manual foi adaptado em uma luva para possibilitar o controle e monitoração do sistema robótico.

Foram conduzidos nove testes que foram realizados em componentes individuais, em sistemas parciais e, no último, foi envolvido o sistema completo. O objetivo dos testes, feitos de forma sequencial até culminar com o sistema completo, foi verificar a robustez das informações comunicadas e processadas utilizando-se o método de ensaios repetitivos. Em todos os testes, verificaram-se respostas comunicadas sem perda de informação e ações de controle (movimento do robô em oito direções, sendo a nona a condição de parada) e monitoração (vibração dos motores de vibração conforme distância) dentro das condições esperadas. No caso das ações, os movimentos seguiram os gestos manuais realizados e as frequências de vibração dos motores ocorreram em função dos intervalos de distância medidos pelos sensores. Em todos os testes, sem exceção, não houve erros observados de comunicação e de movimentos comandados.

Como trabalhos futuros que podem ser conduzidos de forma imediata, a partir do atual estágio implementado neste trabalho, pode-se sugerir:

- a) Aplicar diferentes velocidades de movimento ao robô, a depender da inclinação da mão do usuário;
- b) Aplicar diferentes intensidades de vibração ao *feedback* tátil, a depender das distâncias aos objetos próximos ao robô; e
- c) Troca do sistema de comunicação para o *wifi* de maneira a possibilitar o controle do robô em distâncias maiores;

## REFERÊNCIAS BIBLIOGRÁFICAS

BALLANTYNE, G. H.. Robotic surgery, telerobotic surgery, telepresence and telementoring. **Surgical Endoscopy and other inventional techniques**, Nova York, v. 16, n. 10, p. 1389-1402, jul. 2002. Disponível em: <<http://cmaps.cmapers.net/rid=1HZ2RWKZY-1Y1GHF0-G08/ResearchPaper6.pdf>>. Acesso em: 04 jul. 2017.

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. Disponível em: <[http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/2973/1/CT\\_ENGCOMP\\_2014\\_1\\_04.pdf](http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/2973/1/CT_ENGCOMP_2014_1_04.pdf)>. Acesso em: 06 jul. 2017

BUONOCORE, L. **SLAM em Ambientes Internos Utilizando Robô de Baixo Custo**. Tese de Doutorado em Dispositivos e Sistemas Eletrônicos, Instituto Tecnológico de Aeronáutica, São José dos Campos, 2013.

COUNSELL, M. *Haptic communication for remote mobile and manipulator robot operations in hazardous environments*. Tese de PhD, Universidade de Salford, Reino Unido, 2003. Disponível em: <<http://usir.salford.ac.uk/2039/>>. Acesso em: 02 jul. 2017

GONZATTO, A. et al. Óculos sonar para deficientes visuais. **Encontro latino de Iniciação Científica**, São José dos Campos, v. 13, out. 2009. Disponível em: <[http://www.inicepg.univap.br/cd/INIC\\_2009/anais/arquivos/RE\\_0948\\_0818\\_01.pdf](http://www.inicepg.univap.br/cd/INIC_2009/anais/arquivos/RE_0948_0818_01.pdf)>. Acesso em: 04 jul. 2017.

GUIMARÃES, Vinícius Galvão. **Automação e monitoramento remoto de sistema de irrigação na agricultura**. 2011. 91 f. TCC (Graduação) - Curso de Engenharia Mecatrônica, Universidade de Brasília, Brasília, 2011.

HATZFELD, C.; KERN, T. **Engineering haptic devices** (2nd ed.). Springer: Nova York, 2014. 573 p.

HIRZINGER, G. et al. ROTEX - The first remotely controlled robot in space. **IEEE xplore**, Wessling, v.3, p. 2604 – 2611, mai. 1994. Disponível em: <<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=351121>>. Acesso em: 07 jul. 2017.

JUNIOR, Ary Ferreira Dos Santos; OROZIMBO, Yanne; MARTINS, Lucas. Aplicação de Drones na Logística Humanitária. **Inovarse - Responsabilidade social aplicada**, Rio de Janeiro, v. 3, n. 12, set. 2016. Disponível em: <[http://www.inovarse.org/sites/default/files/T16\\_373.pdf](http://www.inovarse.org/sites/default/files/T16_373.pdf)>. Acesso em: 04 jul. 2017.

NUNES, Urbano José Carreira. **Controlo de robôs com realimentação sensorial no espaço tarefa**. 1995. 297 f. Tese (Doutorado) - Curso de Ciências da Engenharia, Faculdade de Ciências e Tecnologia da Universidade de Coimbra, Coimbra, 1995. Disponível em: <<https://estudogeral.sib.uc.pt/handle/10316/1910>>. Acesso em: 04 jul. 2017.

OKAMURA, Allison M. Haptic feedback in robot-assisted minimally invasive surgery. **Current Opinion in Urology**, Baltimore, v. 19, n. 1, p. 102-107, jan. 2009. Disponível em: <<https://insights.ovid.com/pubmed?pmid=19057225>>. Acesso em: 02 jul. 2017.

PAYNE, C. **Ungrounded haptic-feedback for hand-held surgical robots**. Tese de PhD, Imperial College, Londres, 2015. Disponível em: <<https://spiral.imperial.ac.uk:8443/handle/10044/1/26587>>. Acesso em: 02 jul. 2017.

RIBEIRO, Gabriel Casulari de Motta. **Teleoperação bilateral de múltiplos robôs aplicada ao transporte de carga**. 2013. 141 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2013. Disponível em: <<http://www.pee.ufrj.br/index.php/pt/producao-academica/dissertacoes-de-mestrado/2013-1/2013081901-2013081901/file>>. Acesso em: 04 jul. 2017.

SANTOS Jr., A. F.; OROZIMBO, Y.; MARTINS, L. **Aplicação de Drones na Logística Humanitária**. XII Congresso Nacional de Excelência de Gestão & III Inovarse, 2016. Disponível em: <[http://www.inovarse.org/sites/default/files/T16\\_373.pdf](http://www.inovarse.org/sites/default/files/T16_373.pdf)>. Acesso em: 04 jul. 2017.

THRUN, Sebastian; BURGARD, Wolfram; FOX, Dieter. **Probabilistic Robotics**. Cambridge: The Mit Press, 2006. 647 p.

TOURINO, Sérgio. R. G. **Guiagem do robô móvel para Inspeção de Tubulações Industriais Soldadas via Internet**, Relatório de Projeto de Graduação, 1999. Disponível em: <<http://docslide.com.br/download/link/guiagem-do-robo-movel-xr4000-para-inspecao-de-tubulacoes-industriais>>. Acesso em: 02 jul. 2017.

ZHAI, S.; MILGRAM, P. **Human Robot Synergism and Virtual Telerobotic Control**. Proc. HFAC. Canadá. 1992.

ZILLI, Silvana do Rocio. **A Robótica Educacional no Ensino Fundamental: Perspectivas e Prática**. 89 f, 2004.. Dissertação (Mestrado em Engenharia de Produção) – Programa de Pós-Graduação em Engenharia de Produção, UFSC, Florianópolis. Disponível em: <<https://repositorio.ufsc.br/bitstream/handle/123456789/86930/224814.pdf?sequence>>. Acesso em: 02 jul. 2017.

## APÊNDICE A – Códigos em C/C++ no ATmega328p para realização dos experimentos em componentes e sistema completo

Neste apêndice se encontram todos os códigos utilizados para testes de componentes de ambos os sistemas (controle manual e robô), além dos códigos para os sistemas definitivos.

### A.1 – Código para teste *Bluetooth* transmissor

```
// Teste Bluetooth HC-05 (transmissor)
short int msg;

void setup() {
  Serial.begin(9600); }

void loop() {
  msg=1;
  Serial.println(msg);
  delay(1000);
  msg=0;
  Serial.println(msg);
  delay(1000); }
```

### A.2 – Código para teste *Bluetooth* receptor

```
// Teste Bluetooth HC-05 (receptor)
short int msg, led=13;

void setup() {
  pinMode(led, OUTPUT); //led
  Serial.begin(9600); }

void loop() {
  //espera por mensagem
  while(!Serial.available());

  msg = Serial.read();
  if(msg=='1') {
    digitalWrite(led, HIGH);
    Serial.println("Ligado!"); }
  if(msg=='0'){
    digitalWrite(led, LOW);
    Serial.println("Desligado!"); }
  delay(500); }
```

### A.3 – Código para teste de leitura do acelerômetro (MPU-6050)

```
//teste leitura acelerômetro MPU-6050
#include <Wire.h>

const int MPU_addr=0x68;
int16_t AcX,AcY,AcZ;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
```

```

Wire.write(0x6B); //registrador PWR_MGMT_1
Wire.write(0); //inicia o MPU-6050
Wire.endTransmission();

Serial.begin(9600); }

void loop() {
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B); //ACCEL_XOUT_H
  Wire.endTransmission(false);

  Wire.requestFrom(MPU_addr, 6, true);
  AcX = Wire.read()<<8|Wire.read(); //0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY = Wire.read()<<8|Wire.read(); //0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ = Wire.read()<<8|Wire.read(); //0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)

  Serial.print("AcX = "); Serial.print(AcX);
  Serial.print(" AcY = "); Serial.print(AcY);
  Serial.print(" AcZ = "); Serial.println(AcZ);

  if(AcX>5000)
    digitalWrite(LED_BUILTIN, HIGH);
  else
    digitalWrite(LED_BUILTIN, LOW);

  delay(40); }

```

#### A.4 – Código para teste de leitura do sonar HC-SR04

```

//teste sensor ultrassônico HC-SR04 (sonar)
const int pingPin = 4; //trigger
const int pingReceive = 5; //echo

void setup() {
  Serial.begin(9600);

  pinMode(pingPin, OUTPUT);
  pinMode(pingReceive, INPUT); }

void loop() {
  long duration, cm;

  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin, LOW);

  duration = pulseIn(pingReceive, HIGH);

  cm = duration / 58;

  Serial.print(cm);
  Serial.println(" cm");

  delay(200); }

```

#### A.5 – Código para teste de leitura do sensor infravermelho

```

//teste leitura sensor infravermelho

```



```

int value;
const int pin = A0; //pino do infravermelho

void setup() {
  Serial.begin(9600); }

void loop() {
  value=analogRead(pin);
  Serial.println(value);
  delay(200); }

```

## A.6 – Código para envio da leitura do acelerômetro via *Bluetooth*

```

//envio de dados do acelerômetro via Bluetooth
#include <Wire.h>

const int MPU_addr=0x68;
int16_t AcX,AcY;
short int x,y;

void setup() {
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B); //registrador PWR_MGMT_1
  Wire.write(0); //inicia o MPU-6050
  Wire.endTransmission();

  Serial.begin(9600); }

void loop() {
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B); //ACCEL_XOUT_H
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr, 4, true);
  AcX = Wire.read()<<8|Wire.read(); //0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY = Wire.read()<<8|Wire.read(); //0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)

  //codificação eixo X
  if(AcX>=6000) x=2;
  if(AcX<6000&&AcX>-6000) x=1;
  if(AcX<=-6000) x=0;

  //codificação eixo Y
  if(AcY>=6000) y=5;
  if(AcY<6000&&AcY>-6000) y=4;
  if(AcY<=-6000) y=3;

  //envio dos dados
  Serial.print(x);
  delay(2);
  Serial.print(y);

  delay(100); }

```

## A.7 – Código para recebimento da leitura do acelerômetro via *Bluetooth*

```

//recebimento de dados do acelerômetro via Bluetooth

```

```

short int x,y;

void setup() {
  Serial.begin(9600); }

void loop() {
  while(!Serial.available());
  x = Serial.read();
  while(!Serial.available());
  y = Serial.read();

  //caracteres convertidos para inteiros pela tabela ASCII
  Serial.println(x-48);
  Serial.println(y-48);
  Serial.println("");

  delay(100); }

```

## A.8 – Código para teste dos motores de vibração

```

//teste dos motores de vibração (L293D)
const int motor1pin = 3;
const int motor2pin = 5;
const int motor3pin = 6;

void setup() {
  pinMode(motor1pin, OUTPUT);
  pinMode(motor2pin, OUTPUT);
  pinMode(motor3pin, OUTPUT);

  analogWrite(motor1pin, 0);
  analogWrite(motor2pin, 0);
  analogWrite(motor3pin, 0);

  delay(1000); }

void loop() {
  analogWrite(motor1pin, 140);
  analogWrite(motor2pin, 140);
  analogWrite(motor3pin, 140);
  delay(500);
  analogWrite(motor1pin, 0);
  analogWrite(motor2pin, 0);
  analogWrite(motor3pin, 0);
  delay(500); }

```

## A.9 – Código para teste dos motores do robô móvel

```

//teste de movimento do robô RoSeLi
#include <Wire.h>

//REGISTRADORES
#define MD25ADDRESS      0x58           // Endereço da placa MD-25
#define SPEED1           (byte) 0x00   // Byte para enviar velocidade ao primeiro motor
#define SPEED2           0x01         // Byte para enviar velocidade ao segundo motor

//PROTÓTIPOS
void frente();
void parar();
void direita();

```

```

void esquerda();
void tras();

const int vel = 116; //128 = ponto central, velocidade nula

void setup() {
  Wire.begin();
  Serial.begin(9600); }

void loop() {
  frente(); delay(2000); parar(); delay(200);

  direita(); delay(2000); parar(); delay(200);

  esquerda(); delay(2000); parar(); delay(200);

  tras(); delay(2000); parar(); delay(2200); }

void frente() {
  Wire.beginTransmission(MD25ADDRESS); // Drive do motor 2 na velocidade armazenada em vel
  Wire.write(SPEED2);
  Wire.write(vel);
  Wire.endTransmission();

  Wire.beginTransmission(MD25ADDRESS); // Drive do motor 1 na velocidade armazenada em vel
  Wire.write(SPEED1);
  Wire.write(vel);
  Wire.endTransmission(); }

void parar() {
  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED2);
  Wire.write(128);
  Wire.endTransmission();

  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED1);
  Wire.write(128);
  Wire.endTransmission(); }

void direita() {
  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED2);
  Wire.write(vel);
  Wire.endTransmission();

  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED1);
  Wire.write(256 - vel);
  Wire.endTransmission(); }

void esquerda() {
  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED2);
  Wire.write(256 - vel);
  Wire.endTransmission();

  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED1);
  Wire.write(vel);

```

```

Wire.endTransmission(); }

void tras() {
  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED2);
  Wire.write(256 - vel);
  Wire.endTransmission();

  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED1);
  Wire.write(256 - vel);
  Wire.endTransmission(); }

```

## A.10 – Código para controle de movimento do robô (receptor)

```

//controle de movimento do robô RoSeLi (receptor)
#include <Wire.h>

//REGISTRADORES
#define MD25ADDRESS      0x58
#define SPEED1          (byte) 0x00
#define SPEED2          0x01

//PROTÓTIPOS
void frente(); void parar(); void direita(); void esquerda(); void tras();
void frentedireita(); void frenteesquerda(); void trasdireita(); void trasesquerda();

const int vel = 116; //128 = ponto central, velocidade nula
int x, y, aux;

void setup() {
  Wire.begin();
  Serial.begin(9600); }

void loop() {

  //espera pelos dados do acelerômetro
  while (!Serial.available());
  x = Serial.read();
  while (!Serial.available());
  y = Serial.read();

  if (x > y) {
    aux = x; x = y; y = aux; }

  if (x == '1') {
    if (y == '4') parar();
    if (y == '5') tras();
    if (y == '3') frente(); }

  if (x == '2') {
    if (y == '4') esquerda();
    if (y == '5') trasesquerda();
    if (y == '3') frenteesquerda(); }

  if (x == '0') {
    if (y == '4') direita();
    if (y == '5') trasdireita();
    if (y == '3') frentedireita(); }

```

```

delay(10); }

void frente() {
  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED2);
  Wire.write(vel);
  Wire.endTransmission();

  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED1);
  Wire.write(vel);
  Wire.endTransmission(); }

void parar() {
  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED2);
  Wire.write(128);
  Wire.endTransmission();

  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED1);
  Wire.write(128);
  Wire.endTransmission(); }

void direita() {
  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED2);
  Wire.write(vel);
  Wire.endTransmission();

  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED1);
  Wire.write(256 - vel);
  Wire.endTransmission(); }

void esquerda() {
  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED2);
  Wire.write(256 - vel);
  Wire.endTransmission();

  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED1);
  Wire.write(vel);
  Wire.endTransmission(); }

void tras() {
  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED2);
  Wire.write(256 - vel);
  Wire.endTransmission();

  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED1);
  Wire.write(256 - vel);
  Wire.endTransmission(); }

void frentedireita() {
  Wire.beginTransmission(MD25ADDRESS);
  Wire.write(SPEED2);

```

```

Wire.write(vel);
Wire.endTransmission();

Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED1);
Wire.write((vel + 128) / 2);
Wire.endTransmission(); }

void frenteesquerda() {
Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED2);
Wire.write((vel + 128) / 2);
Wire.endTransmission();

Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED1);
Wire.write(vel);
Wire.endTransmission(); }

void trasdireita() {
Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED2);
Wire.write(256 - vel);
Wire.endTransmission();

Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED1);
Wire.write(256 - (vel + 128) / 2);
Wire.endTransmission(); }

void trasesquerda() {
Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED2);
Wire.write(256 - (vel + 128) / 2);
Wire.endTransmission();

Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED1);
Wire.write(256 - vel);
Wire.endTransmission(); }

```

## A.11 – Código para o controlador manual

```

//CONTROLADOR MANUAL
#include <Wire.h>

//PROTÓTIPOS
void ordenar(void); //ordena leitura recebida para casos de problema na comunicação

//MOTORES VIBRACAO (L293D)
const int enable1 = 8;
const int enable2 = 9;

const int motor1pin = 3;
const int motor2pin = 5;
const int motor3pin = 6;

const int pwm_forte = 180;
const int pwm_fraco = 100;

```

```

//ACELEROMETRO
const int MPU_addr = 0x68;
int16_t AcX, AcY;
short int x, y, z, aux;

void setup() {
  //SETUP MOTORES VIBRACAO
  pinMode(enable1, OUTPUT);
  pinMode(enable2, OUTPUT);
  pinMode(motor1pin, OUTPUT);
  pinMode(motor2pin, OUTPUT);
  pinMode(motor3pin, OUTPUT);

  analogWrite(motor1pin, 0);
  analogWrite(motor2pin, 0);
  analogWrite(motor3pin, 0);

  //SETUP ACELEROMETRO
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B); //registrador PWR_MGMT_1
  Wire.write(0); //inicia o MPU-6050
  Wire.endTransmission();

  Serial.begin(9600);

  //tempo de espera para os módulos Bluetooth se conectarem
  delay(5000); }

void loop() {
  //LEITURA ACELEROMETRO
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B); //ACCEL_XOUT_H
  Wire.endTransmission(false);

  Wire.requestFrom(MPU_addr, 4, true);
  AcX = Wire.read() << 8 | Wire.read(); //0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY = Wire.read() << 8 | Wire.read(); //0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)

  //ENVIAR DADOS DO ACELEROMETRO
  if (AcX >= 6000) x = 2;
  if (AcX < 6000 && AcX > -6000) x = 1;
  if (AcX <= -6000) x = 0;

  if (AcY >= 6000) y = 5;
  if (AcY < 6000 && AcY > -6000) y = 4;
  if (AcY <= -6000) y = 3;

  Serial.print(x);
  delay(2);
  Serial.print(y);
  delay(2);

  //RECEBER DADOS DOS SENSORES
  while (!Serial.available()); //123
  x = Serial.read();
  while (!Serial.available()); //456
  y = Serial.read();
  while (!Serial.available()); //789
  z = Serial.read();

```

```
ordenar();

//ACIONAMENTO DA VIBRAÇÃO
if (x == '1') analogWrite(motor2pin, 0);
if (x == '2') analogWrite(motor2pin, pwm_fraco);
if (x == '3') analogWrite(motor2pin, pwm_forte);
if (y == '4') analogWrite(motor1pin, 0);
if (y == '5') analogWrite(motor1pin, pwm_fraco);
if (y == '6') analogWrite(motor1pin, pwm_forte);
if (z == '7') analogWrite(motor3pin, 0);
if (z == '8') analogWrite(motor3pin, pwm_fraco);
if (z == '9') analogWrite(motor3pin, pwm_forte); }
```

```
void ordenar() {
  if (x > y) {
    aux = x; x = y; y = aux; }
  if (x > z) {
    aux = x; x = z; z = aux; }
  if (y > z) {
    aux = y; y = z; z = aux; }
  if (x > y) {
    aux = x; x = y; y = aux; }
  if (x > z) {
    aux = x; x = z; z = aux; }
  if (y > z) {
    aux = y; y = z; z = aux; }
}
```

## A.12 – Código para o robô móvel

```
//CODIGO FINAL ROBÔ MÓVEL
#include <Wire.h>

//REGISTRADORES
#define MD25ADDRESS      0x58      // Endereço da placa MD25
#define SPEED1           (byte) 0x00 // Byte para enviar velocidade para o primeiro motor
#define SPEED2           0x01      // Byte para enviar velocidade para o segundo motor

//PROTÓTIPOS
void frente();
void parar();
void direita();
void esquerda();
void tras();
void frentedireita();
void frenteesquerda();
void trasdireita();
void trasesquerda();

//VARIÁVEIS
const int vel = 116; //128 = ponto central, velocidade nula
const int infrared = A0; //pino do infravermelho
const int echo1 = 3, echo2 = 5;
const int trig1 = 2, trig2 = 4;
int leitura1, leitura2, leitura3;
int x, y, aux;

void setup() {
```



```

pinMode(13, OUTPUT);
Wire.begin();
Serial.begin(9600);

pinMode(trig1, OUTPUT);
pinMode(echo1, INPUT);
pinMode(trig2, OUTPUT);
pinMode(echo2, INPUT); }

void loop() {
//RECEBER DADOS DO ACELEROMETRO
while (!Serial.available());
x = Serial.read();
while (!Serial.available());
y = Serial.read();

if (x > y) {
  aux = x; x = y; y = aux; }

//MOVIMENTAÇÃO DO ROBÔ
if (x == '1') {
  if (y == '4') parar();
  if (y == '5') direita();
  if (y == '3') esquerda(); }

if (x == '2') {
  if (y == '4') frente();
  if (y == '5') frentedireita();
  if (y == '3') freteesquerda(); }

if (x == '0') {
  if (y == '4') tras();
  if (y == '5') trasdireita();
  if (y == '3') trasesquerda(); }

delay(1);

//LEITURA DOS SENSORES E ENVIO
leitura1 = map(analogRead(infrared), 0, 600, 80, 0); //valormax = 558, max=80cm

digitalWrite(trig1, LOW);
delayMicroseconds(2);
digitalWrite(trig1, HIGH);
delayMicroseconds(5);
digitalWrite(trig1, LOW);
leitura2 = (pulseIn(echo1, HIGH)) / 58;

digitalWrite(trig2, LOW);
delayMicroseconds(2);
digitalWrite(trig2, HIGH);
delayMicroseconds(5);
digitalWrite(trig2, LOW);
leitura3 = (pulseIn(echo2, HIGH)) / 58;

if (leitura1 > 25) leitura1 = 1;
else if (leitura1 > 15) leitura1 = 2;
else leitura1 = 3;

if (leitura2 > 25) leitura2 = 4;
else if (leitura2 > 15) leitura2 = 5;

```

```

else leitura2 = 6;

if (leitura3 > 20) leitura3 = 7;
else if (leitura3 >= 15) leitura3 = 8;
else leitura3 = 9;

Serial.print(leitura1);
delay(2);
Serial.print(leitura2);
delay(2);
Serial.print (leitura3);
delay(2);
}

void frente() {
Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED2);
Wire.write(vel);
Wire.endTransmission();

Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED1);
Wire.write(vel);
Wire.endTransmission(); }

void parar() {
Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED2);
Wire.write(128);
Wire.endTransmission();

Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED1);
Wire.write(128);
Wire.endTransmission(); }

void direita() {
Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED2);
Wire.write(vel);
Wire.endTransmission();

Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED1);
Wire.write(256 - vel);
Wire.endTransmission(); }

void esquerda() {
Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED2);
Wire.write(256 - vel);
Wire.endTransmission();

Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED1);
Wire.write(vel);
Wire.endTransmission(); }

void tras() {
Wire.beginTransmission(MD25ADDRESS);

```

```

Wire.write(SPEED2);
Wire.write(256 - vel);
Wire.endTransmission();

Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED1);
Wire.write(256 - vel);
Wire.endTransmission(); }

void frentedireita() {
Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED2);
Wire.write(vel);
Wire.endTransmission();

Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED1);
Wire.write((vel + 128) / 2);
Wire.endTransmission(); }

void frenteesquerda() {
Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED2);
Wire.write((vel + 128) / 2);
Wire.endTransmission();

Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED1);
Wire.write(vel);
Wire.endTransmission(); }

void trasdireita() {
Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED2);
Wire.write(256 - vel);
Wire.endTransmission();

Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED1);
Wire.write(256 - (vel + 128) / 2);
Wire.endTransmission(); }

void trasesquerda() {
Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED2);
Wire.write(256 - (vel + 128) / 2);
Wire.endTransmission();

Wire.beginTransmission(MD25ADDRESS);
Wire.write(SPEED1);
Wire.write(256 - vel);
Wire.endTransmission(); }

```