CENTRO PAULA SOUZA
GOVERNO DO ESTADO DE SÃO PAULO

Faculdade de Tecnologia de Americana
Curso Superior de Tecnologia em Desenvolvimento de Jogos Digitais

UNIDADE ROBÓTICA MÓVEL COM ARDUINO CONTROLADA POR UMA APLICAÇÃO ANDROID VIA BLUETOOTH

GUSTAVO BERNO

Americana, SP
2014

CENTRO PAULA SOUZA
GOVERNO DO ESTADO DE SÃO PAULO

Faculdade de Tecnologia de Americana
Curso Superior de Tecnologia em Desenvolvimento de Jogos Digitais

UNIDADE ROBÓTICA MÓVEL COM ARDUINO CONTROLADA POR UMA APLICAÇÃO ANDROID VIA BLUETOOTH

GUSTAVO BERNO

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Desenvolvimento de Jogos Digitais, sob a orientação do Prof. Me. Clerivaldo José Roccia.

Área: Jogos Digitais

Americana, SP
2014

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

Berno, Gustavo
B448u Unidade robótica móvel com arduino controlada por uma aplicação android via bluetooth. / Gustavo Berno. – Americana: 2014.
47f.

Monografia (Graduação de Tecnologia em Jogos Digitais). - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza.
Orientador: Prof. Me. Clerivaldo José Roccia

1.Robótica I. Roccia, Clerivaldo José II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana.

CDU: 007.52

GUSTAVO BERNO

UNIDADE ROBÓTICA MÓVEL COM ARDUINO CONTROLADA POR UMA APLICAÇÃO ANDROID VIA BLUETOOTH

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Tecnologia em Desenvolvimento de Jogos Digitais pelo CEETEPS/Faculdade de Tecnologia – FATEC/ Americana.

Área: Jogos Digitais.

Americana, 01 de Julho de 2014.

Banca Examinadora:

_____
Prof. Me. Clerivaldo José Roccia (Presidente)

_____
Prof. Dra. Acácia De Fátima Ventura (Membro)

_____
Prof. Gustavo Carvalho Gomes De Abreu (Membro)

AGRADECIMENTOS

Primeiramente, gostaria de agradecer aos meus pais, Terezinha e Oscar, minhas irmãs Ludimila e Ludinéia, que com muito amor e carinho me auxiliaram nessa jornada, sempre acreditando em mim e no meu potencial. Por todas as horas de conversas e desabafos, com palavras suaves para me animar. Sou muito grato por tudo.

Também agradeço ao meu orientador Me. Clerivaldo José Roccia por toda a atenção e colaboração no desenvolvimento desse trabalho.

Aos meus amigos de república (Rep du Jack), e à Bruna Goulart, Fernando Oliveira, Ivo Perruci Neto, Camila Molly, Elen Perez, Willian Fuertes, Rodrigo Campagnoli, entre outros amigos e colegas de classe que me deram apoio, meus sinceros agradecimentos também.

A todos, meu muito obrigado!

DEDICATÓRIA

Dedico este trabalho à minha grande fonte de motivação,
minha família.

“Existem muitas hipóteses em ciência que estão erradas. Isso é perfeitamente aceitável, eles são a abertura para achar as que estão certas”.

Carl Sagan

RESUMO

Com a facilidade que temos atualmente para desenvolver plataformas robóticas e independentes, existe cada vez mais a necessidade de comunicação entre diferentes interfaces. Motivado pelo grande crescimento do uso da plataforma *Arduino* na área da robótica para a prototipação de novas ideias, atrelado ao crescimento do uso do sistema operacional *Android* nos dispositivos móveis em todo o mundo. O objetivo deste trabalho foi construir uma plataforma robótica móvel com a plataforma *Arduino* e desenvolver um sistema Android capaz de manipulá-la e comandá-la através de sinais digitais emitidos pelas interfaces de comunicação *Bluetooth* usando um *smartphone*. O sistema desenvolvido permite o controle de um robô móvel de pequeno porte, com movimentos de avançar, retroceder e girar para ambos os lados.

Palavras chave: *Arduino, Android, Bluetooth,* Robótica.

ABSTRACT

With the facility that we currently have to develop independent robotic platforms, there is increasingly a need for communication between different interfaces. Motivated by the increased use of the Arduino platform in robotics for prototyping new ideas, linked with the growing use of the Android operating system on mobile devices worldwide. The objective of this work was to build a mobile robotic platform with the Arduino platform and develop an Android system, able to manipulate it and command it via digital signals emitted by Bluetooth communication interfaces using a smartphone. The developed system allows the control of a small mobile robot with movements forward, backward and turn to both sides.

Keywords: *Arduino*, *Android*, *Bluetooth*, Robotics.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

1 INTRODUÇÃO

O tema Robótica Móvel ganha mais relevância quando a literatura científica que trata do assunto tem destacado e garantido o quão importante o desenvolvimento de pesquisa na área para o futuro. Prevê-se que a Robótica Móvel autônoma será responsável por uma nova revolução no mundo moderno, trazendo inúmeros benefícios e grandes desafios para um futuro próximo (MARTINS, 2006).

A utilização de plataformas robóticas móveis tem crescido nas mais diversas áreas tais como industrial, científica, médica e ensino (MARTINS, 2006; WARREN, 2011). Envolvendo e desenvolvendo o conhecimento em áreas da eletrônica, computação e mecânica que, integrados de forma concisa, possibilitam a criação de uma plataforma robótica multipropósito que interage e percebe o ambiente real através de seus atuadores. Através de um microcontrolador é exercido o gerenciamento e o funcionamento de todos os dispositivos ligados a ele, garantindo a sincronia necessária para o funcionamento de todos os elementos (TAVARES, 2013).

Atualmente, existem no mercado, algumas plataformas robóticas móveis tais como a Plataforma *Arduino (*MASSIMO, 2008*)* e a Plataforma *RaspBerry Pi (*RASPBERRY, 2014*)*. Em todas essas plataformas, alguns recursos são comuns tais como: possui um microcontrolador, uma unidade de memória, capacidade de processamento limitada, capacidade de interação com o meio ambiente através dos atuadores e sensores e capacidade de comunicação sem fios usando tecnologias como *Bluetooth* e *Wi-Fi* (WARREN, 2011).

A disponibilidade de diversas bibliotecas de software com várias funções já implementadas para essas plataformas robóticas facilita o desenvolvimento de projetos robóticos com grande facilidade e rapidez.

Aliado ao desenvolvimento das plataformas robóticas, os dispositivos móveis produzidos pela indústria tais como *Smartphones, Tablets e Notebooks,* possuem também, em sua maioria uma interface para a comunicação sem fios direta, através

da interface *Bluetooth*. Isso possibilita o controle dos dispositivos robóticos de forma remota e confiável.

Outro ponto relevante que deve ser considerado está relacionado à evolução dos sistemas operacionais para dispositivos móveis tais como *Android, IOS* e *Windows Phone.* Esses sistemas operacionais permitem a construção de aplicativos de forma rápida e confiável. Além dos sistemas operacionais, também podemos citar o desenvolvimento das linguagens de programação voltadas para os dispositivos móveis tais como JAVA, Object C e C# (PEREIRA; SILVA, 2009; ANDROID, 2014b).

Como posse deste conjunto de recursos (sistema operacional, linguagem de programação e plataforma robótica), é possível o desenvolvimento de uma infinidade de soluções para as mais diversas aplicações.

É possível realizar a comunicação entre uma plataforma robótica com *Arduino* através de uma aplicação *Android* de forma segura e confiável? A ideia é criar um aplicativo para qualquer sistema operacional *Android* que possua um módulo *Bluetooth* no qual este possa ser utilizado para controlar qualquer plataforma robótica que utilize a plataforma *Arduino*, apenas implementando um código de recepção dos dados na plataforma.

O objetivo geral deste trabalho é obter uma comunicação estável entre o *Arduino* e *Android* via comunicação *Bluetooh*, confiável e eficiente.

Para tanto, os objetivos específicos são:

a) Utilizar ferramentas livres, como *software* e *hardware open source*, para garantir o baixo custo do projeto;
b) Adquirir conhecimento na plataforma *Arduino*;
c) Desenvolver uma aplicação com *Android* controlando remotamente via *Bluetooth* uma unidade robótica móvel equipada com *Arduino;*

O trabalho se justifica pela viabilidade deste projeto, tornando-o possível e relevante, visando às determinadas hipóteses:

a) Ser um assunto atual e relevante;
b) Desenvolver conhecimentos aplicáveis na comunicação entre dispositivos de tecnologias distintas, criando um projeto eficiente e de baixo custo;

c) Adquirir conhecimentos em plataformas robóticas utilizando o conhecimento computacional desenvolvido ao longo do curso;

Neste Trabalho é detalhado o desenvolvimento da plataforma robótica móvel, desde a plataforma *Arduino* até a comunicação com o *Android*. O capitulo 2 descreve a caracterização de uma plataforma robótica. O capitulo 3 descreve a implementação de todo o contexto do projeto. O capitulo 4 demonstra os resultados e discussões da plataforma robótica e sua relação entre o aplicativo móvel. O capítulo 5 traz as conclusões acerca do desenvolvimento. Os apêndices A, B e C contem o esquemático da plataforma, código do *Arduino* e o código do *Android* respectivamente.

2 METODOLOGIA

Neste capítulo serão apresentadas as principais características de uma plataforma robótica móvel.

2.1 Robótica

A Robótica é uma ciência que mescla física, mecânica, matemática, design, inteligência artificial, eletrônica e programação, e tem por objetivo compreender o processo de controle e montagem de "sistemas que interagem com o mundo real com pouca ou mesmo nenhuma intervenção humana" (MARTINS, 2006). Os sistemas que processam informações para efetuar ações são chamados de robôs.

Com o surgimento da necessidade de desenvolver sistemas cada vez mais complexos, assim como os que sempre foram retratados em filmes de ficção cientifica - e estes foram grandes impulsionadores de ideias para que fossem desenvolvidas no mundo real - muitas das tarefas antes executadas por seres-humanos hoje são substituídas por aplicações robôs, se não por completo ao menos parte desta tarefa, como operações que requerem precisão, esforço e agilidade (WARREN, 2011).

2.2 Plataforma Robótica

A plataforma robótica móvel é composta por um *Arduino UNO* com microcontrolador ATMega328, uma Ponte H *Motor Shield* com chip L928, um modulo *Bluetooth* JY-MCU HC-06 para a comunicação, dois motores *DC*(*Direct Curent)*[1] e as baterias necessárias.

Para a plataforma de controle foi utilizado um smartphone *Samsung S Duos* com sistema operativo *Android* e comunicação *Bluetooth*.

A seguir são apresentadas e detalhadas as funcionalidades de cada item citado anteriormente a ser utilizado na montagem da plataforma robótica móvel.

[1] DC - Direct Curent (Corrente contínua)

2.3 Arduino

Arduino é uma plataforma *open-source* de prototipagem eletrônica flexível e fácil de usar. É destinado a artistas, designers, entusiastas e qualquer pessoa interessada em criar objetos ou ambientes interativos. (ARDUINO, 2014).

A plataforma envolve um modelo distribuído de desenvolvimento de *hardware* com os contribuintes de diferentes partes do mundo. Diferente de sistemas fechados, projetos *open-source* permitem uma liberdade individual para acessar os arquivos de origem de um projeto, fazer melhorias, e redistribuir essas melhorias para uma comunidade maior (EVANS, 2011). O modelo utilizado na plataforma robótica móvel é o *Arduino UNO*, que pode ser adquirida facilmente através da internet ou em lojas de eletrônica e robótica. (Figura 1).

Figura 1 - Placa Arduino UNO

Fonte: (ARDUINO,2014)

O Arduino UNO utiliza o microcontrolador ATMega328 da Atmel, que utiliza a arquitetura de Harvard e é de 8 bits, RISC[2], com 32KB de memória *flash*, 1KB de EEPROM[3], 2KB de SRAM[4], 32 registradores de uso geral, 3 temporizadores/contadores, uma USART[5], portas para comunicação SPI[6], 6

[2] RISC: "*Reduced Instruction Set Computer"* ou "Computador com conjunto reduzido de instruções"
[3] EEPROM: "*Electrically-Erasable Programmable Read-Only Memory"* é um tipo de memória não volátil e que pode ser reescrita um número limite de vezes.
[4] SRAM: "*Static Random Access Memory"* é um tipo de memória volátil de acesso randômico.
[5] USART: "Universal Synchronous/Asynchronous Receiver/Transmitter" é utilizado para realizar a comunicação serial com o microcontrolador.
[6] SPI: "Serial Peripheral Interface" é utilizado para implementar comunicação serial com o Microcontrolador.

conversores AD[7] de 10bits e um *watchdog timer*[8], entre outras características. A tensão de operação dele é entre 1,8 e 5,5 V (ATMEL, 2014).

O *software* que permite desenvolver códigos e lógica de programação para que a plataforma *Arduino possa definir suas ações é o* Arduino *IDE* (*Integrated Development Environment*), que aceita uma linguagem de programação simples e intuitiva chamada *Processing* (PROCESSING, 2014), e quando este código é compilado a *IDE* o traduz para uma linguagem entendida pelo microcontrolador (Linguagem C). Este último passo é muito importante, porque é onde Arduino torna a sua vida simples, escondendo o máximo possível das complexidades da programação de microcontroladores (MASSIMO, 2008).

Codificar e desenvolver projetos com *Arduino* é simples, pois necessita um conhecimento basico de programação e não necessáriamente um conhecimento avançado em eletronica. Há duas funções para programar que necessitam ser declaradas: a função **setup()** para configurar o estado dos pinos de entrada e saída, bem como outras ações que só precisam ser executados uma vez, e a função **loop()** irá realizar quaisquer ações dentro dela repetidamente. Cada linha de código é executada dentro de uma função, em ordem, a partir do topo para a base. Dentro do **loop(),** quando a última linha de código é alcançado, a função será repetida, começando de novo com a primeira linha do circuito da função **loop()** (EVANS, 2011).

A plataforma *Arduino* foi desenvolvida para permitir a expansão de suas funcionalidades de acordo com a necessidade ou a complexidade do projeto. As funções da plataforma podem ser ampliadas com a adição de periféricos conhecidos como *shields* que podem ser acoplados diretamente na placa ou interligadas através de uma placa de ensaio conhecida como *protoboard*.

2.4 Android OS

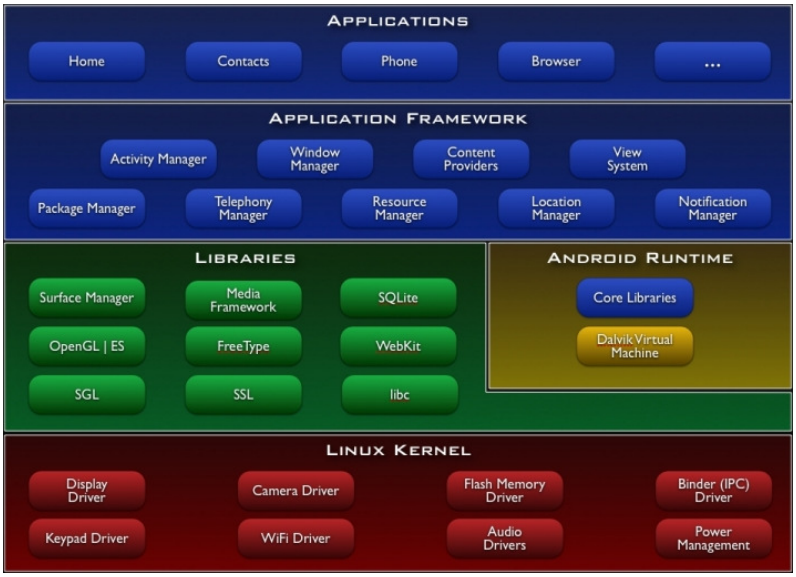
Android é um sistema operacional *open-source* para dispositivos móveis que é baseado em uma versão modificada do Linux. A principal vantagem de adotar o *Android* é que ele oferece uma abordagem unificada para o desenvolvimento de

[7] AD: Analógico Digital
[8] Utilizado para forçar um reset no microcontrolar, reiniciando seu programa.

aplicativos. Os desenvolvedores precisam apenas desenvolver para o *Android*, e suas aplicações são capazes de funcionar em vários diferentes dispositivos móveis, desde que os dispositivos tenham o sistema operacional. No mundo dos smartphones, os aplicativos são a parte mais importante da cadeia de sucesso (LEE, 2012).

O *Android* possui uma arquitetura dividida em cinco componentes. Aplicações, *Framework* de Aplicação, Bibliotecas, Ambiente de Execução e o *Kernel* do Linux, como mostrada na Figura 2. (ANDROID, 2014a)

Figura 2 – Esquemático da Arquitetura do *Android*.



Fonte: (ANDROID, 2014a).

Onde:

- Camada *Applications* (Aplicações): No qual estão os aplicativos nativos do Android que incluem aplicativos de e-mails, calendário, navegador de internet, programa SMS (*Short Message Service*), mapa GPS (*Global Positioning System*), entre outros.

- Camada *Application Framework* (Framework de aplicação): Apresenta os componentes que permitirão que novas estruturas sejam utilizadas para o desenvolvimento de novas aplicações.

- Camada de *Libraries* (bibliotecas): São bibliotecas padrão do Android escrita em C/C++. Também possui as bibliotecas de multimídia, do acelerômetro, banco de dados (*SQLigth*) e fontes bitmap.

- Camada Android *Runtime* (ambiente de execução): É a máquina virtual *Dalvik* responsável pela execução dos códigos Java das aplicações.

- *Kernel Linux* 2.6 (*Kernel* do Linux): Fornece os serviços do núcleo do sistema. (ANDROID, 2014b).

2.5 Periférico de comunicação

2.5.1 Bluetooth

O *Bluetooth* é um protocolo padrão de comunicação sem fio projetado para baixo consumo de energia, com baixo alcance (BLUETOOTH, 2014), baseado em microchips transmissores de baixo custo com o propósito de eliminar as conexões físicas entre dispositivos de diferentes fabricantes (MILLER, 2001).

Os principais serviços oferecidos por esta tecnologia são: transferência de arquivos entre computadores, transferência de dados entre computadores e impressoras e transferência de áudio entre telefones e fones de ouvidos sem fio.

Existem três classes do dispositivo *Bluetooth*, onde a diferença entre elas está na potência do sinal e alcance (BLUETOOTH, 2014). Sendo:

Classe 1 potência de 100 mW (20 dBm) e alcance de até 100 metros;
Classe 2 com potência de 2,5 mW (4dBm) e alcance de até 10 metros;
Classe 3 com 1 mW (0dBm) de potência e 1 metro de alcance.

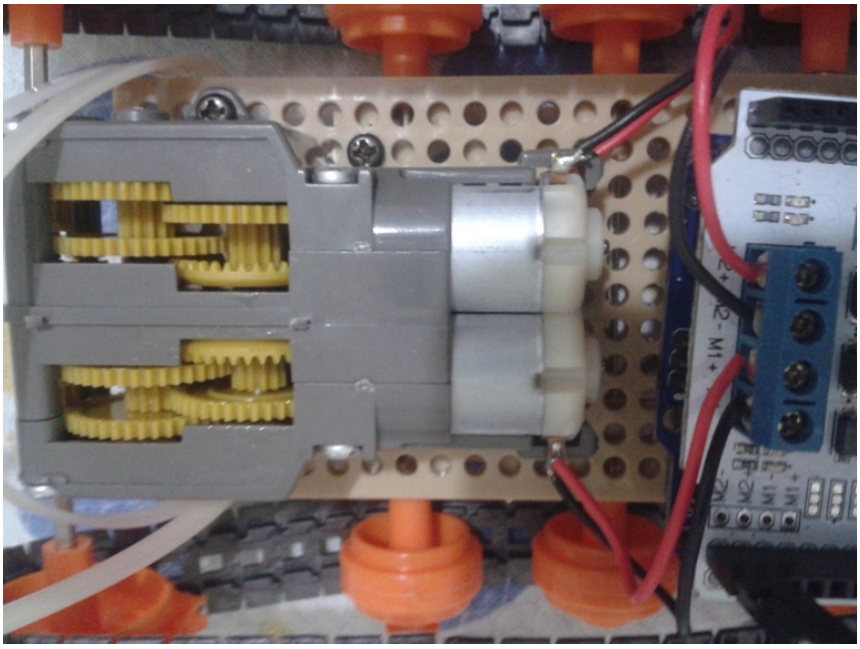
2.6 Atuador

2.6.1 Motores DC

Motores *DC* estão em praticamente qualquer dispositivo que tenha movimento como: reprodutores de video cassete, toca-fitas, brinquedos, eletronicos com movimentos rotativos, entre outros. Por existir um uso comum deste dispositivo é facil encontra-lo ou remove-lo de algum dispositivo eletronico, como mostrado anteriormente. Como eles são raramente soldado a uma placa de circuito impresso, então você pode simplesmente desconectar os fios e remover qualquer parte que o prenda. Uma vez removido, você pode testar o motor por ligá-la com uma bateria entre 6v a 12v, dependendo do tamanho do motor (WARREN, 2011).

A plataforma possui dois motores DC com caixas de redução acopladas aos eixos para realizar o movimento das rodas do robô (Figura 3).

Figura 3 – Motores DC



Fonte: elaborado pelo autor

Tabela 1 – Especificações do motor

Típica tensão de operação	3V
Opções de redução	58:1 e 204:1
Velocidade do motor sem carga @ 3V	12300 rpm
Corrente do motor sem carga @ 3V: 15	150mA
Corrente de Stall @ 3V	2100mA
Torque do motor @ 3V	0,5 oz*in

Fonte: (ROBOCORE, 2014)

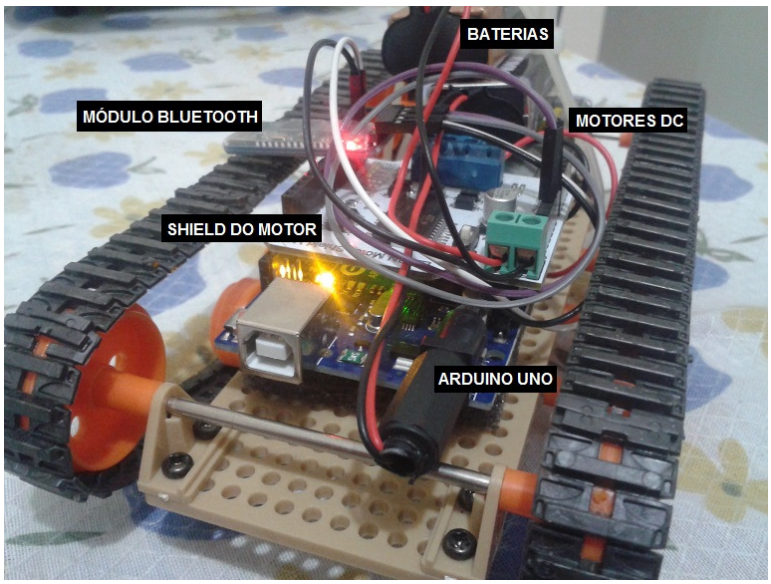
3 IMPLEMENTAÇÃO

Neste capitulo, serão demonstrados os passos e integrações necessárias para o desenvolvimento deste projeto.

3.1 Robô

A plataforma robótica móvel foi desenvolvida utilizando o *Arduino UNO* com um microcontrolador ATMega328 (ATMEL, 2014), a qual coordena todos os eventos que ocorrem com a plataforma. Na Figura 4 temos a localização dos componentes.

Figura 4 – Localização das peças na plataforma



Fonte: elaborado pelo autor

Deste modo o *Arduino UNO* consegue gerenciar todos os periféricos ligados a ele e assim possibilitando o veiculo robotizado a realizar as seguintes funções:

- Andar para frente;
- Andar para trás;
- Girar no eixo para direita;
- Girar no eixo para esquerda;
- interromper;

- Comunicar-se via *Bluetooth*;

O código estruturado para esse robô, em sua essência, é um *loop* infinito onde é realizada uma leitura do *Bluetooth* (JY-MCU, 2014), e com base nessas informações são acionados seus atuadores.

3.2 Plataforma de controle

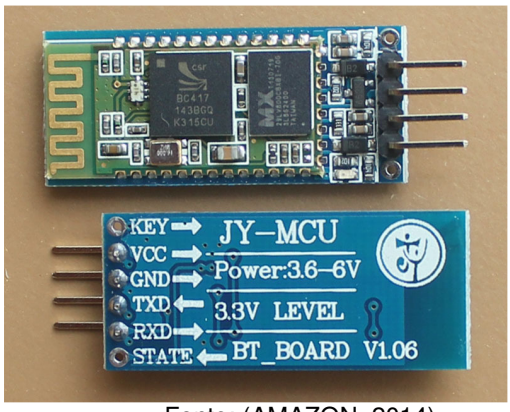
3.2.1 Controle através do Android

A plataforma criada pode ser controlada também através de um Smartphone, utilizando o sistema operacional *Android*.

Para isso, foi criado um pequeno programa, possibilitando o controle do robô com o uso de uma interface do tipo *touchscreen* (pressionando botões na tela do celular).

Para estabelecer conexão entre o Arduino e o *smarthphone*, fez-se necessária a instalação de um módulo Bluetooth no sistema (vide Figura 5). Este módulo é independente e tem como função receber os comandos do aplicativo Android através da interface Bluetooth e repassá-los via comunicação serial para o Arduino, que, por sua vez, recebe os comandos do módulo, interpreta-os e controla os motores (MONK, 2010).

Figura 5 - O módulo Bluetooth utilizado é fabricado pela empresa Lson.



Fonte: (AMAZON, 2014)

O Bluetooth apresentado na figura 5 é de classe 2, portanto a comunicação entre da plataforma se limita a 10 metros de distancia (JY-MCU, 2014).

Figura 6 – Comunicação entre o *Android* e o *Arduino*.

Fonte: (ANDRUX, 2014)

A Figura 6 mostra uma referência de conexão do módulo *Bluetooth* ao microcontrolador *Arduino UNO*.

O *smartphone* com *Android* rodando o aplicativo se comunica exclusivamente com o módulo Bluetooth, assim como a comunicação do *Arduino* também é feita exclusivamente com o módulo Bluetooth, e este é o único que possui comunicação com ambos os lados. Ou seja, não existe comunicação direta entre o *Arduino* e o *Android*.

3.1 Comandos e decisões

3.1.1 Comandos

Os comandos implementados para a plataforma são mostrados na tabela 2.

Tabela 2 – Comandos da plataforma.

Estado	Comando Enviado
FRENTE	'F'
ESQUERDA	'L'
DIREITA	'R'
RÉ	'B'
PARADO	'P'

Fonte: elaborado pelo autor

3.1.2 Decisões

Descrevendo os principais eventos realizados pela plataforma quando um novo comando é recebido. As entradas e como elas interagem com a dinâmica da plataforma são mostrados a seguir:

Andar para frente

Quando o robô recebe o comando de andar para frente, as seguintes ações internas são realizadas:

- Os pinos referentes ao sentido de rotação de ambos os motores da plataforma são ajustados para que os dois motores rodem em sentido horário, propulsionando o robô pra frente.

Essa condição é mantida enquanto o botão “frente” da aplicação continuar sendo pressionado.

Andar para trás

Quando o robô recebe o comando de andar para trás, as seguintes ações internas são realizadas:

- Os pinos referentes ao sentido de rotação de ambos os motores da plataforma são ajustados para que os dois motores rodem em sentido anti-horário, propulsionando o robô pra trás.

Essa condição é mantida enquanto o botão “trás” da aplicação continuar sendo pressionado.

Girar para esquerda

Quando o robô recebe o comando de girar para esquerda, as seguintes ações internas são realizadas:

- Os pinos referentes ao sentido de rotação do motor A é ajustado para que o mesmo rode no sentido horário, e o motor B no sentido anti-horário.

Essa condição é mantida enquanto o botão “esquerdo” da aplicação continuar sendo pressionado.

Girar para direita

Quando o robô recebe o comando de girar para direita, as seguintes ações internas são realizadas:

- Os pinos referentes ao sentido de rotação do motor A é ajustado para que o mesmo rode no sentido anti-horário, e o motor B no sentido horário.

Essa condição é mantida enquanto o botão “direito” da aplicação continuar sendo pressionado.

Parar

O robô só recebe esta instrução quando algum dos botões anteriores for solto, fazendo a plataforma parar.

- Os pinos de PWM dos motores são ajustados para zero, independentemente do sentido dos motores.

Essa condição é mantida enquanto um novo comando não for recebido.

3.2 Software do robô

O *software* criado foi desenvolvido na ferramenta *Arduino IDE*, com os métodos e operações necessários para coordenar todos os periféricos do robô. Como veremos a seguir.

3.2.1 Arduino

Contém os métodos ***setup()*** e ***loop().*** Durante o método ***setup()*** são realizadas as chamadas dos métodos necessários para inicializar o controle da plataforma: Inicialização da comunicação Bluetooth. Se nenhum problema for detectado, o *led (Light Emitting Diode)*[9] vermelho no módulo fica aceso. Caso

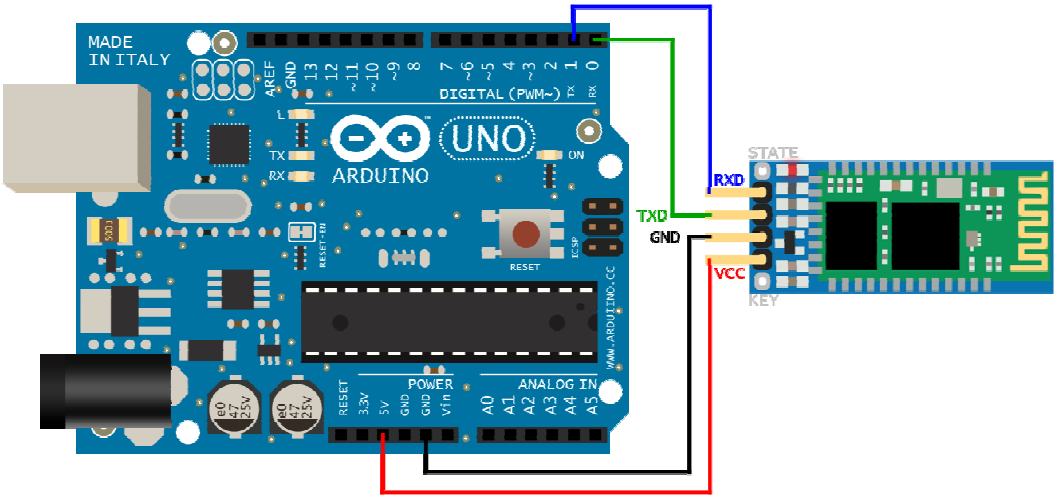
[9] LED - **L**ight **E**mitting **D**iode (Diodo emissor de luz).

contrário, o *led* permanece piscando indicando que não foi possível iniciar a comunicação (EVANS, 2011).

O método ***loop()*** realiza a leitura dos dados do controle e envia o comando recebido para a plataforma robótica.

Como apresentado o *Arduino* tem uma ligação Bluetooth para a comunicação com o dispositivo *Android* móvel, assim demonstrado na figura 7.

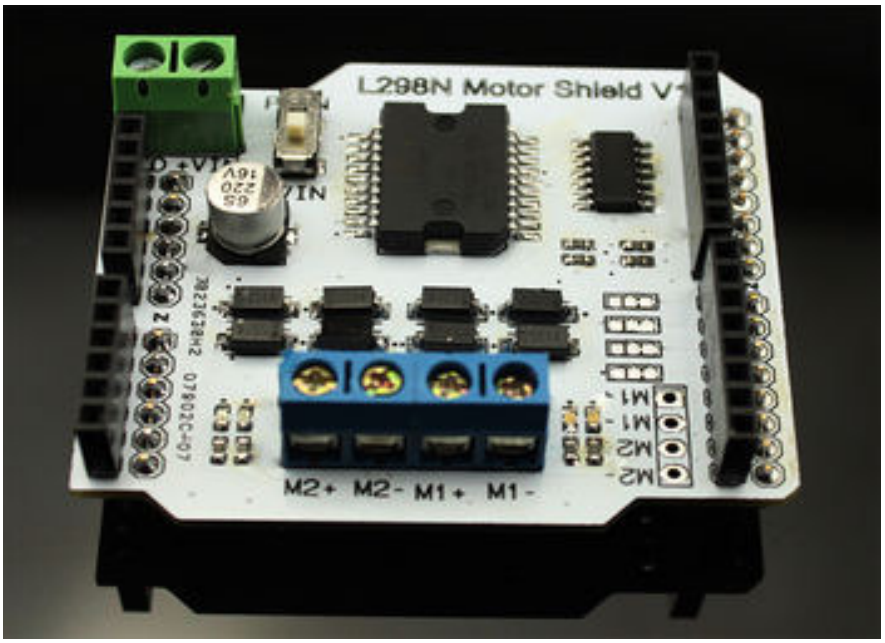
Figura 7 – Ligação na plataforma *Arduino UNO*.



Fonte: (JY-MCU,2014)

E para os controles dos Motores DC foi adicionada uma Shield de Ponte-H (WARREN, 2011) L298 como mostra a figura 8 e suas características técnicas na Tabela 3 ligada ao Arduino, que após receber os comandos via *Bluetooth* executa as determinadas tarefas pré-definidas.

Figura 8 – Motor Shield L298



Fonte: (MOTOR SHIELD, 2014)

Tabela 3 – Especificação do Motor Shield

Voltagem de Controle Lógico	5V (Fornecido pelo Arduino)
Voltagem do Drive do Motor	6.5~12v(Alimentação pino VIN), 4.8~35V (Alimentação Externa)
vias para controle de motores	2
Corrente de Alimentação Lógica ISS	≤36mA
Corrente do Drive Motor IO	≤2A
Consumo máximo de força	25W （T=75℃）
Corrente em cada via	Até 2A
Pinos usados para conduzir os motore	Pinos 4, 5, 6 e 7
Controle de velocidade	PWM ou PLL

Fonte: (MOTOR SHIELD, 2014)

3.3 Software de controle

Para que uma conexão *Bluetooth* seja estabelecida entre *Android* e *Arduino*, é necessário que seja incluso no código o endereço físico do dispositivo que receberá a conexão, neste caso o módulo *Bluetooth* que estará conectado ao *Arduino*. Após a seleção do módulo *Bluetooth* para conexão, será possível controlar a plataforma robótica como apresentada na seção 3.5.1 (MONK, 2010).

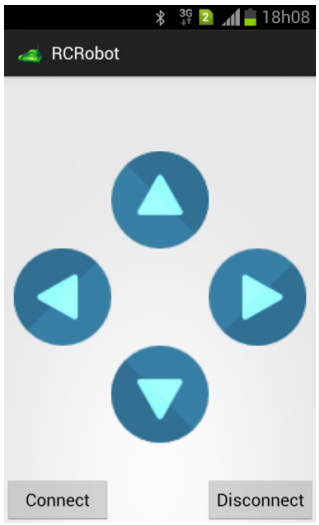
3.3.1 Aplicativo RCRobot

Trata-se da implantação da comunicação com a plataforma robótica *Arduino* utilizando o B*luetooth* como protocolo de comunicação.

O aplicativo *Android* é constituído por duas telas, uma é considerada a principal e denomina-se “MainActivity”. A “MainActivity” é onde estão localizados os controles dos movimentos do robô. Enquanto a outra tela tem o objetivo de pesquisar os dispositivos externos para a conexão e está designada como “*Connection*”.

Se ao abrir o aplicativo *Bluetooth* do celular estiver desligado, o aplicativo pedirá a solicitação ao usuário para a ativação do *Bluetooth*. Por outro lado, no momento em que o aplicativo for inicializado e o celular estiver com o *Bluetooth* ligado não aparecerá a solicitação, e sim a tela “MainActivity” que será aberta normalmente como ilustrado na Figura 9.

Figura 9 – Tela Principal MainActivity.



Fonte: elaborado pelo autor

Depois que a aplicação estiver em sua tela principal “MainActivity” com o *Bluetooth* já ativado será necessária a conexão com o dispositivo externo. O botão “Connect” da tela principal (MainActivity) deverá ser “clicado” e assim a tela de pesquisa (Connection) será chamada.

Figura 10 – Tela de conexão (Connection).

Fonte: elaborado pelo autor

Conforme apresentado na Figura 10, a tela de pesquisa (*Connection*) apresentará uma lista com os dispositivos externos encontrados em um raio de 10 metros e um botão para refazer a pesquisa.

O momento em que a tela de pesquisa é chamada, ela ativará o modo de pesquisa do rádio *Bluetooth* do celular Android ocupando toda a banda de conexão e armazenará na lista do “Connection” os nomes dos dispositivos encontrados e seus respectivos endereços *MAC (Media Access Control*).

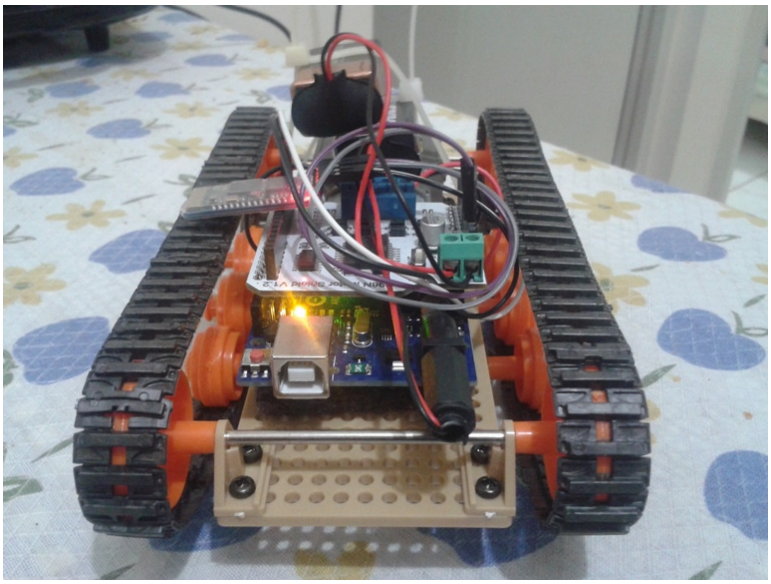
O botão “*Search Device*” demonstrado na figura 10 tem a função de fazer a pesquisa por dispositivos externos. Depois que o dispositivo é encontrado, neste caso a plataforma, o item onde se encontra o nome do dispositivo deve ser clicado e assim o endereço do dispositivo é retornado para a tela “MainActivity”.

Os dados enviados para o robô são um caractere ou um valor inteiro, definindo a trajetória da plataforma. Os caracteres são ‘F’ (frente), ‘L’ (esquerda), ‘R’ (direita), ‘B’ (trás) e ‘P’ (parado). O robô já tem sua programação definida esperando apenas o caractere para executar seu movimento. Assim, os movimentos do robô são controlados pelo *smartphone*.

Para finalizar a conexão basta clicar no botão “Disconnect” localizado na tela principal do aplicativo.

Desta forma é possível controlar remotamente todos os comandos da plataforma robótica (figura 11).

Figura 11 – Plataforma Robótica móvel finalizada.



Fonte: elaborado pelo autor

4 RESULTADOS E DISCUSSÕES

O desenvolvimento desse trabalho teve inicio com alguns testes em bancada para descobrir como se realizava o controle de motores DC utilizando o *Arduino* e a *Shield* do motor e também foi realizado testes de comunicação do módulo *Bluetooth*.

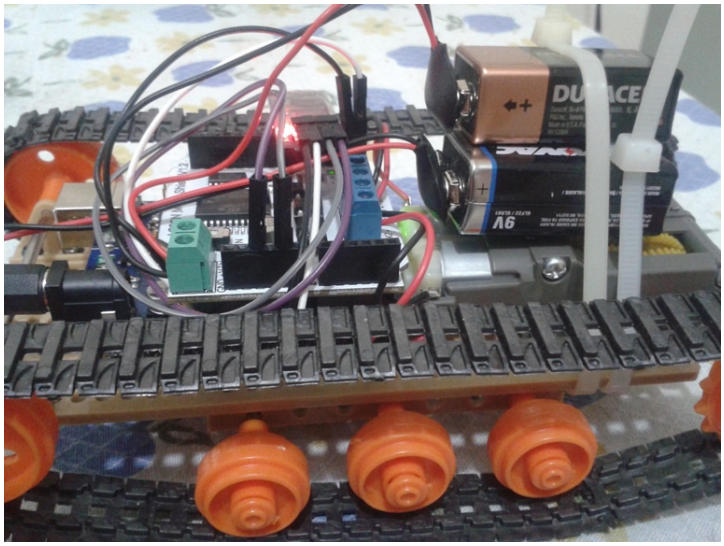
O circuito de Ponte H atualizando o CI L298N (MOTOR SHIELD, 2014) funciona bem no acionamento dos motores, permitindo a mobilidade do mesmo, controlando os motores DC. Porém após os testes realizados apresentou problemas referentes à "*cache*", pois em alguns momentos o excesso de comandos causava o estouro da memória do cache o que ocasionava a parada do envio de sinal. Como este não foi o foco do trabalho, demostra-se apenas como sendo uma falha da "*shield*".

O aplicativo desenvolvido para o *Android* não apresentou problemas de comunicação, execução ou falhas lógicas no sistema.

O sistema de comunicação *Bluetooth,* funciona corretamente, porém utilizando a tecnologia *Bluetooth* a área de alcance do link de comunicação entre o robô e o celular é pequeno, restrito a dez metros, permitindo um *link* de comunicação segura. Testes em ambientes com objetos e sem objetos entre o dispositivo móvel e a plataforma robótica móvel, demonstram interferências no sinal de comunicação e variações da distancia especificada, entre 7 e 10 metros.

As "esteiras" que são responsáveis pela movimentação (figura 12) da plataforma apresentam fragilidade e dependendo do terreno no qual o robô se movimenta e as sequencias de movimentações realizadas, muitas vezes acabam saindo do encaixe, necessitando a recolocação das mesmas. Assim, este periférico demonstra que sua funcionalidade é apenas didática, não podendo ser aplicada para fins industriais ou de uso continuo.

Figura 12 – Periférico de movimentação (esteiras) da plataforma.



Fonte: elaborado pelo autor

A plataforma robótica não possui um controle preciso dos motores, assim não é possível indicar uma quantidade fixa de graus para o robô movimentar sobre o próprio eixo, ou garantir que ele fique paralelo a uma reta enquanto anda para frente ou para trás. Apesar de não ter sido o objetivo o uso, um motor de passo poderia se adequar a solucionar este problema já que este tem um controle mais preciso.

Como a disposição deste trabalho não visa demonstrar tecnicamente a montagem desta plataforma, apenas a análise e seus resultados, foi disponibilizado um tutorial passo a passo deste Projeto. (BERNO, 2014c).

5 CONCLUSÃO

Para o desenvolvimento desta plataforma robótica móvel foi necessário compreender as necessidades de Hardware, Software, mecânica e eletrônica, para o qual o conjunto final realizasse a tarefa desejada.

Com a popularização da plataforma *Arduino* e do sistema operacional móvel Android, é possível encontrar muitas referências boas, tais como: Livros, fóruns na internet, trabalhos acadêmicos entre outras fontes de informação. Contudo, a biblioteca *Bluetooth* do *Android* foi um obstáculo que decorreu da bibliografia disponível. As bibliografias pesquisadas apresentam um conteúdo muito superficial sobre esse assunto. Neste caso, a solução encontrada foi pesquisar o material bibliográfico de referência do portal de desenvolvedores oficial da *Google*.

Sendo assim, se tornou pratico e viável demonstrar e aprender com a interação entre o Android e a robótica móvel através de uma conexão sem fio *Bluetooth*, pois permite, a um baixo custo, demonstrar a abrangência e integração de várias tecnologias envolvendo Engenharia da Computação e a Engenharia de Controle e Automação. Haja vista que várias tecnologias computacionais estão atuando de forma conjunta, tais como, linguagem de programação Java com o sistema Android, comunicação por meio de *Bluetooth* e a abrangência da plataforma Arduino com o microcrontrolador ATMega328.

Foi construída uma plataforma robótica que envolveu diferentes áreas do conhecimento, o que a torna uma excelente plataforma de aprendizagem e introdução para o mundo da robótica e suas possibilidades.

Realizar essa integração demonstrou ainda que utilizar robôs educacionais e empregar desafios como a integração das tecnologias, e reflete o conhecimento obtido durante o curso de graduação. A robótica acaba sendo uma ótima maneira de introduzir esse universo de conhecimento de uma maneira prática, funcional e didática. Agregando assim grande valor técnico, teórico e cientifico obtido com o desenvolvimento dessa plataforma robótica.

Algumas dificuldades foram encontradas no início e após o desenvolvimento do projeto especificamente no que tange a qualidade da plataforma de movimentação

adquirida, dificuldade no controle preciso dos motores, entre outras dificuldades que impediu um fluxo constante e crescente de evolução do projeto, porém, apesar de todos estes problemas a aplicação e a plataforma mostram-se confiável quanto ao seu objetivo.

O desafio de desenvolver um aplicativo que possibilita a comunicação via *Bluetooth* com a plataforma *Arduino* foi superado com este trabalho. Seu desenvolvimento mostra-se alinhado aos objetivos do projeto com grande possibilidade de expansão futura.

Os testes de estabilidade da conexão e alcance da comunicação comprovam que o aplicativo pode ser utilizado com segurança em novos projetos da área. A movimentação dos motores e da plataforma pode ser melhorada através da aquisição de melhores peças e componentes.

Para trabalhos futuros a plataforma foi construída de forma modular, permitindo a expansão de novos recursos. Facilitando assim, futuros projetos relacionados a este.

REFERÊNCIAS

AMAZON, **JY-MCU Arduino Bluetooth Wireless**, Disponível em: <http://www.amazon.com/JY-MCU-Arduino-Bluetooth-Wireless-Serial/dp/B009DZQ4MG/ref=sr_1_6?ie=UTF8&qid=1406729497&sr=8-6&keywords=bluetooth+arduino>. Acesso em 21 mai. 2014.

ANDROID. **Architecture**, Disponível em: <https://source.android.com/devices/graphics/architecture.html>. Acesso em: 20 mar. 2014.

ANDROID. **Getting Started**, Disponível em: <http://developer.android.com/training/index.html>. Acesso em: 20 mar. 2014.

ANDRUX. **Requerimientos de la app**, Disponível em: <http://andrux.com.mx/apps/ArduiBot/index.php/site/index>. Acesso em: 20 mar. 2014.

ARDUINO. **Geting Started - Introduction**, Disponível em: <http://arduino.cc/en/Guide/Introduction>. Acesso em: 02 abr. 2014

ATMEL. **ATmega328**, Disponível em: <http://www.atmel.com/pt/br/devices/atmega328.aspx>. Acesso em: 02 abr. 2014.

BERNO, Gustavo. **Código *Android* de controle da plataforma robótica**. Disponível em: <https://github.com/gberno/RCRobot_Android>. Acesso em: 02 jun. 2014.

BERNO, Gustavo. **Código *Arduino* da plataforma robótica**. Disponível em: <https://github.com/gberno/Arduino_Robot>. Acesso em: 02 jun. 2014.

BERNO, Gustavo. **Tutorial – Como construir uma plataforma robótica móvel com Arduino**. Disponível em: <https://github.com/gberno/Tutorial_Arduino>. Acesso em: 25 jun. 2014.

BLUETOOTH. **Bluetooth Basics**, Disponível em: <http://www.bluetooth.com/Pages/Basics.aspx>. Acesso em: 25 mar. 2014.

EVANS, Brian. **Beginning Arduino Programming**. Estados Unidos: Apress, 2011.

JY-MCU. Disponível em: <http://www.tuxti.com.br/wiki/index.php/Arduino_-_Bluetooth_jy-mcu_HC-06>. Acesso em: 25 mar. 2014.

LEE, Wei-Meng. **Android 4, Application development**. Estados Unidos: Editora: John Wiley & Sons, 2012.

MARTINS, Agenor. **O Que é Robótica**. São Paulo: Editora Brasiliense, 2006.

MASSIMO, Banzi. **Getting Started with Arduino**. Estados Unidos: O'Reilly Media, 2008.

MILLER, Michael. **Descobrindo Bluetooth**. Rio de Janeiro: Editora: Campus, 2001.

MONK, Simon. **30 Arduino Projects for the Evil Genius**. Estados Unidos: McGraw-Hill, 2010.

MOTOR SHIELD. Disponível em: <http://www.tuxti.com.br/wiki/index.php/Arduino_-_L298_Motor_Shield>. Acesso em: 25 mar. 2014.

PEREIRA, Lucio Camilo Oliveira; SILVA, Michel Lourenço da. **Android para Desenvolvedores**. Rio de Janeiro: Brasport, 2009.

PROCESSING. **Home page**, Disponível em: <http://www.processing.org/>. Acesso em 21 mai. 2014.

RASPBERRY PI. **Home page**, Disponível em: <http://www.raspberrypi.org/>. Acesso em 21 mai. 2014.

TAVARES, Luíz Antônio. Uma solução com Arduino para controlar e monitorar processos industriais. **Revista Controle & Instrumentação**, São Paulo, n. 185, abr. 2013.

WARREN, John, et al. **Arduino Robotics**. Estados Unidos: Apress, 2011.

APÊNDICE A – CÓDIGO FONTE DO ROBÔ (ARDUINO)

```
//Declaração dos pinos do motor
int E1 = 5;
int M1 = 4;
int E2 = 6;
int M2 = 7;
char rec;

void setup(){
  pinMode(M1, OUTPUT);
  pinMode(M2, OUTPUT);
  pinMode(E1, OUTPUT);
  pinMode(E2, OUTPUT);
  Serial.begin(9600);//Iniciando comunicação Serial
}

void loop(){
  if(Serial.available()){
    rec = Serial.read();
    if(rec == 'F')front();
    if(rec == 'L')left();
    if(rec == 'R')right();
    if(rec == 'B')back();
    if(rec == 'S')stoprobot();
  }
  Serial.flush();
}

void stoprobot(){
  digitalWrite(E1, LOW);
  digitalWrite(E2, LOW);
  digitalWrite(M1, LOW);
  digitalWrite(M2, LOW);
}

void front(){
  digitalWrite(E1, HIGH);
  digitalWrite(M2, HIGH);
  digitalWrite(E2, HIGH);
}

void back(){
  digitalWrite(E2, HIGH);
  digitalWrite(M1, HIGH);
  digitalWrite(E1, HIGH);
}
void left(){
  digitalWrite(E1, HIGH);
```

```
  digitalWrite(E2, HIGH);
}
void right(){
  digitalWrite(M1, HIGH);
  digitalWrite(E1, HIGH);
  digitalWrite(M2, HIGH);
  digitalWrite(E2, HIGH);
}
```

APÊNDICE B – CÓDIGO FONTE DA PLATAFORMA DE CONTROLE (ANDROID)

MainActivity.java

```
package br.com.example.rcrobot;
import java.io.DataOutputStream;
import java.io.IOException;
import java.util.UUID;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.Menu;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnTouchListener;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.Toast;

public class MainActivity extends Activity {
	private BluetoothAdapter adapter;
	private BluetoothDevice device;
	private BluetoothSocket socket;
	private DataOutputStream output;
	private Intent intent;
	private static final int Connection = 1;
	private String address;
	private ConnectThread OpenConn;

	//Botões
	private Button btConnect;
	private Button btDisconnect;
	private ImageButton btFront;
	private ImageButton btLeft;
	private ImageButton btRight;
	private ImageButton btBack;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);

		device = null;
```

```
        address = null;
        intent = new Intent(this, Connection.class);
        btConnect = (Button) findViewById(R.id.btConnect);
        btConnect.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                        startActivityForResult(intent, Connection);}
        });

        btDisconnect = (Button) findViewById(R.id.btDisconnect);
        btDisconnect.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                        OpenConn.cancel();}
        });

        /*============== Configurar FRONT ==============*/
        btFront = (ImageButton) findViewById(R.id.btFront);
        btFront.setOnTouchListener(new OnTouchListener() {
                @Override
                public boolean onTouch(View v, MotionEvent event) {
                        if (device == null) {
                                alerta("Not connected");
                        } else if (device != null) {
                                switch (event.getAction()) {
                                        case MotionEvent.ACTION_DOWN:

btFront.setImageResource(R.drawable.front_on);
                                                new Send('F').start();
                                        break;
                                        case MotionEvent.ACTION_UP:
btFront.setImageResource(R.drawable.front_off);
                                                new Send('S').start();
                                }
                        }
                        return true;
                }
        });
        /*============== Configurar LEFT ==============*/
        btLeft = (ImageButton) findViewById(R.id.btLeft);
        btLeft.setOnTouchListener(new OnTouchListener() {
                @Override
                public boolean onTouch(View v, MotionEvent event) {
                        if (device == null) {
                                alerta("Not connected");
                        } else if (device != null) {
                                switch (event.getAction()) {
                                        case MotionEvent.ACTION_DOWN:
btLeft.setImageResource(R.drawable.left_on);
                                                new Send('L').start();
```

```
                                        break;
                                        case MotionEvent.ACTION_UP:
btLeft.setImageResource(R.drawable.left_off);
                                                new Send('S').start();
                                }
                        }
                        return true;
                }
        });
        /*============== Configurar RIGHT ==============*/
        btRight = (ImageButton) findViewById(R.id.btRight);
        btRight.setOnTouchListener(new OnTouchListener() {
                @Override
                public boolean onTouch(View v, MotionEvent event) {
                        if (device == null) {
                                alerta("Sem dispositivo Conectado");
                        } else if (device != null) {
                                switch (event.getAction()) {
                                        case MotionEvent.ACTION_DOWN:
btRight.setImageResource(R.drawable.right_on);
                                                new Send('R').start();
                                        break;
                                        case MotionEvent.ACTION_UP:
btRight.setImageResource(R.drawable.right_off);
                                                new Send('S').start();
                                }
                        }
                        return true;
                }
        });
        /*============== Configurar BACK ==============*/
        btBack = (ImageButton) findViewById(R.id.btBack);
        btBack.setOnTouchListener(new OnTouchListener() {
                @Override
                public boolean onTouch(View v, MotionEvent event) {
                        if (device == null) {
                                alerta("Sem dispositivo Conectado");
                        } else if (device != null) {
                                switch (event.getAction()) {
                                        case MotionEvent.ACTION_DOWN:
btBack.setImageResource(R.drawable.back_on);
                                                new Send('B').start();
                                        break;
                                        case MotionEvent.ACTION_UP:
btBack.setImageResource(R.drawable.back_off);
                                                new Send('S').start();
                                }
                        }
                        return true;
                }
```

```
		});

		/*Confirando a comunicação bluetooth */
		adapter = BluetoothAdapter.getDefaultAdapter();

		if (!adapter.isEnabled()) {
			Intent enableBtIntent = new Intent(
					BluetoothAdapter.ACTION_REQUEST_ENABLE);
			startActivityForResult(enableBtIntent, 1);
		}
	}

	private class ConnectThread extends Thread {
		private final BluetoothDevice mDevice;

		public ConnectThread(BluetoothDevice device) {
			mDevice = device;
			BluetoothSocket tmp = null;
			try {
				tmp =
device.createRfcommSocketToServiceRecord(UUID
						.fromString("00001101-0000-1000-8000-
00805F9B34FB"));
			} catch (IOException e) {
				e.printStackTrace();
			}
			socket = tmp;
		}

		public void run() {
			adapter.cancelDiscovery();
			try {
				socket.connect();
				alerta("Connected");
			} catch (IOException e) {
				try {
					socket.close();
					alerta("Connection Error");
				} catch (IOException e1) {
					e1.printStackTrace();
				}
				return;
			}
		}

		public void cancel() {
			try {
				socket.close();
				alerta("End Connection");
			} catch (IOException e) {
```

```
                        e.printStackTrace();
                    }
                }
            }

            protected void onActivityResult(int cod, int result, Intent intent) {
                if (intent == null) {
                    alerta("Nothing Selected");
                    return;
                }

                address = intent.getExtras().getString("msg");
                alerta("New device " + address);

                device = adapter.getRemoteDevice(address);
                OpenConn = new ConnectThread(device);
                OpenConn.start();
            }

            public class Send extends Thread {
                private char letter;

                public Send(char letter) {
                    this.letter = letter;
                }

                public void run() {
                    try {
                        output = new
DataOutputStream(socket.getOutputStream());
                        output.writeChar(letter);
                        output.flush();

                    } catch (IOException erro) {
                        alerta("Transfer Error");
                    }
                }
            }


            @Override
            public boolean onCreateOptionsMenu(Menu menu) {
                getMenuInflater().inflate(R.menu.activity_main, menu);
                return true;
            }

            private final Handler hand = new Handler() {
                public void handleMessage(Message msg) {
                    String content = (String) msg.obj;
```

```
                        Toast.makeText(MainActivity.this, content,
Toast.LENGTH_SHORT)
                                        .show();
                }
        };

        public void alerta(String message) {
                Message mes = hand.obtainMessage();
                mes.obj = message;
                hand.sendMessage(mes);
        }
}
```

Connection.java

```
package br.com.example.rcrobot;

import java.util.Set;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;

public class Connection extends Activity {
        private BluetoothAdapter adapt;
        private ListView list;
        private ArrayAdapter<String> dev;
        private Button btSearch;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.connection);
```

```
		adapt = BluetoothAdapter.getDefaultAdapter();
		btSearch = (Button) findViewById(R.id.btSearch);

		btSearch.setOnClickListener(new OnClickListener() {
			@Override
			public void onClick(View v) {
				search();
			}
		});

		list = (ListView) findViewById(R.id.lsList);
		dev = new ArrayAdapter<String>(this,
				android.R.layout.simple_expandable_list_item_1);
		list.setAdapter(dev);
		list.setOnItemClickListener(new OnItemClickListener() {
			@Override
			public void onItemClick(AdapterView<?> arg0, View view, int
position,
					long id) {
				String info = ((TextView) view).getText().toString();
				String address = info.substring(info.length() - 17);


				Intent it = new Intent();
				it.putExtra("msg", address);
				setResult(Activity.RESULT_OK, it);
				finish();
			}
		});

		IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
		this.registerReceiver(mReceiver, filter);

		filter = new
IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
		this.registerReceiver(mReceiver, filter);

		Set<BluetoothDevice> pairedDevices = adapt.getBondedDevices();

		if (pairedDevices.size() > 0) {
			for (BluetoothDevice device : pairedDevices) {
				dev.add(device.getName() + "\n"+ device.getAddress());
			}
		}
	}

	private void search() {
		alert("Searching ... ");
		if (adapt.isDiscovering()) {
```

```
                adapt.cancelDiscovery();
            }

            adapt.startDiscovery();
            dev.clear();
        }

        @Override
    protected void onDestroy() {
        super.onDestroy();
        if (adapt != null) {
          adapt.cancelDiscovery();
        }
        this.unregisterReceiver(mReceiver);
    }

        private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
            public void onReceive(Context context, Intent intent) {
                String action = intent.getAction();
                if (BluetoothDevice.ACTION_FOUND.equals(action)) {
                    BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                    dev.add(device.getName() + "\n"+ device.getAddress());
                }
            }
        };

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            getMenuInflater().inflate(R.menu.activity_main, menu);
            return true;
        }

        private final Handler h = new Handler() {
            public void handleMessage(Message msg) {
                String content = (String) msg.obj;
                Toast.makeText(Connection.this, content,
Toast.LENGTH_SHORT).show();
            }
        };

        public void alert(String message) {
            Message mes = h.obtainMessage();
            mes.obj = message;
            h.sendMessage(mes);
        }
}
```