CENTRO UNIVERSITÁRIO DE BRASÍLIA -UniCEUB

CURSO DE ENGENHARIA DE COMPUTAÇÃO

BRUNO PEREIRA PASSOS

SISTEMA *BLUETOOTH* PARA CONTROLE DE ACESSÓRIOS VEICULARES UTILIZANDO SMARTPHONE COM ANDROID

Orientadora: M.C. Maria Marony Sousa Farias

Brasília

Junho, 2011

BRUNO PEREIRA PASSOS

SISTEMA *BLUETOOTH* PARA CONTROLE DE ACESSÓRIOS VEICULARES UTILIZANDO SMARTPHONE COM ANDROID

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientadora: M.C.Maria Marony Sousa Farias

Brasília

Junho, 2011

BRUNO PEREIRA PASSOS

SISTEMA *BLUETOOTH* PARA CONTROLE DE ACESSÓRIOS VEICULARES UTILIZANDO SMARTPHONE COM ANDROID

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientadora: M.C.Maria Marony Sousa Farias

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS.

_____

Prof. Abiézer Amarília Fernandez
Coordenador do Curso

Banca Examinadora:

_____

Prof. Maria Marony Sousa Farias, M.C.em Engenharia Elétrica
Orientadora

_____

Prof. José Julimá Bezerra Júnior, MSc.
Instituto Militar de Engenharia – IME/RJ

_____

Prof. Francisco Javier de Obaldía, MSc.
Instituição

AGRADECIMENTOS

Inicialmente gostaria de agradecer a Deus por ter me dado tudo o que tenho até hoje e de maneira especial, a minha família que sempre foi à base de tudo e que sempre estiveram ao meu lado, me dando apoio nos momentos difíceis e celebrando a cada sucesso obtido. Sem eles esse projeto não teria sido possível.

Agradeço também ao meu tio Robert que foi a pessoa da qual o projeto foi originado e de maneira vital foi um participante ativo para a construção e desenvolvimento do mesmo. Algumas contribuições foram muito importantes para o funcionamento do protótipo e na escrita da monografia como a que me foi dada pelos amigos de classe: Gabriel Santos, Guilherme Costa, Samyr Alves, André Luiz e Rafael Rennó.

À professora Maria Marony que foi a minha orientadora e ao professor Francisco Javier deixo o meu muito obrigado porque foram imprescindíveis durante a implementação do projeto.

Para Mônica Soares minha namorada, eu só tenho que agradecer pela paciência que teve durante esse primeiro semestre de 2011.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este projeto consiste na construção de um protótipo para simular o controle de algumas funcionalidades automotivas, com a utilização de um Smartphone com Android sem a necessidade da utilização da chave de ignição do próprio veículo. O objetivo dessa simulação não está relacionado à substituição da chave automotiva e sim com a geração de uma nova opção para a utilização do celular no controle de funções automotivas. O protótipo possui diversos servomotores que representam o funcionamento de vidros e travas elétricas, interligados a um microcontrolador Arduino e a um módulo *Bluetooth* que fornece o meio de comunicação com o dispositivo móvel. Um *software* chamado Amarino está instalado no Smartphone. E este, faz o uso do acelerômetro do celular, e dependendo do movimento realizado no celular uma instrução é enviada para a interpretação. O Arduino contém a programação das funções a serem realizadas pelo protótipo. O Smartphone é utilizado no envio e no recebimento de informações correspondentes ao funcionamento das atividades. O *Software Livre* Amarino é a aplicação responsável pelo estabelecimento da conexão, definição do evento e verificação das atividades realizadas pelo protótipo.

Palavras Chave: Android, Arduino, Amarino, Servomotor, Smartphone, *Bluetooth*.

ABSTRACT

This project consists in the construction of a prototype to simulate the control of some automotive functionalities, with the utilization of a Smartphone with Android, without the need of the utilization of the ignition key of the vehicle itself. The objective of this simulation is not related to the replacement of the automotive's key, but with the creation of a new option for the utilization of the cell phone in the control of automotive functions. The prototype has several servomotors that represent the behavior of the windows and power door locks, interlinked to a microcontroller Arduino and a Bluetooth's module that provides the route of communication with the mobile device. A software called Amarino is installed on the Smartphone. And this, makes use of the accelerometer, and depending of the movement performed on the mobile an instruction is send to interpretation. The Arduino contains the programmation of the functions to be accomplished by the prototype. The Smartphone is in charge of the despatch and the reception of the instructions corresponding to the execution of the activities. The Free Software Amarino is the application responsible for the establishment of the connection, definition of the event and verification of the activities accomplished by the prototype.

Keys Words: Android, Arduino, Amarino, Servomotor, Smartphone, *Bluetooth*.

1. INTRODUÇÃO

1.1. Apresentação do Problema

É muito comum a chave de ignição apresentar defeito ou ser esquecida no interior do veículo. Caso ocorra uma das situações anteriores, pode haver o travamento das portas e vidros, e com isso, gerar um problema de abertura do veiculo. Ainda existe possibilidade de problemas de comunicação com o celular, tais como: falta de sinal ou a comunicação não está funcionando de forma correta. Com base nos problemas descritos, este projeto se baseia na criação de um protótipo com utilização de um Smartphone com Android para a realização dos destravamento de portas e vidros elétricos, para ter acesso ao interior do veículo.

1.2. Objetivos do Trabalho

O objetivo geral deste trabalho é construir um protótipo que simule os vidros e travas elétricas de um veículo. Para representar tais funcionalidades foi necessário utilizar servomotores. O microcontrolador utilizado é o Arduino e a comunicação com o celular é feita por conexão *Bluetooth*. É utilizado o aplicativo Amarino instalado em um Smartphone com Android para controle das funcionalidades do protótipo.

Levando em consideração o problema descrito no item 1.1 e no objetivo geral mencionado acima, ficam definidos alguns itens a serem desenvolvidos e testados no decorrer desse projeto:

- Desenvolvimento do código para Arduino contendo cada funcionalidade a ser executada no protótipo. Na plataforma Arduino a linguagem utilizada é o Wiring. Com isso, é sendo possível receber os eventos provenientes dos diversos sensores existentes no Smartphone. Após os dados serem recebidos e com os processamentos devidamente realizados, a funcionalidade adequada é então executada.
- Montagem do protótipo com os servomotores representando de forma simples as funcionalidades reais do automóvel, ou seja, conexão de cabos de energia e dados, servomotores e placa de acrílico (vidros) e trava automotiva da porta (trava) para representar a simulação dos itens de um carro.
- Configuração do dispositivo *Bluetooth*, tanto o que estará interligado ao protótipo, quanto o do dispositivo celular, como definição de nomes dos dispositivos, senhas, proteções de acesso como bloqueio por número de MAC e outros itens dependendo da viabilidade do mesmo.

- Instalação da aplicação open source Amarino no Smartphone com Android e configuração de eventos para que seja possível transmitir os dados para o dispositivo Arduino existente no protótipo, além de configurações de segurança do dispositivo *Bluetooth*.
- Calibração dos valores enviados, afim de que o tratamento posterior possa ser realizado com maior facilidade e independente da opção a ser realizada a mesma possa estar em um padrão aceitável. Nessa etapa inúmeros testes serão realizados para comprovar o real funcionamento e com isso, melhorias poderão ser identificadas ou problemas que podem ser solucionados ou não.
- A customização da aplicação Amarino poderá ocorrer levando em consideração os problemas e falhas encontrados no decorrer da implementação, e também poderá ser necessária caso as funcionalidades não estejam sendo realizadas de forma correta, caso o tempo para a criação seja viável, a fim de que melhore algumas etapas para que o objetivo geral do projeto possa ser cumprido.

1.3. Justificativa e Importância do Trabalho

O problema descrito no item 1.1 é um fato corriqueiro, e é frustrante não poder ter acesso ao interior do veículo por estar em um lugar afastado das vias urbanas ou mesmo em uma viagem para outro estado da federação e sem nenhum meio de comunicação ativa, como um simples sinal de celular. Esse é o ponto de partida desse projeto. Outro fato importante é que o carro pode não ter uma chave de ignição reserva. Baseado nesse fato e no anteriormente descrito é que surgem as três hipóteses para a solução do problema:

- A primeira consiste em arrombamento do veículo, nesse caso existe o prejuízo que ocorre pelo dano de algum item do veículo, seja o vidro, trava da porta ou qualquer outro problema que possa vir a surgir, com base nisso é um fato que deve ser desconsiderado (ou considerado em último caso);
- A segunda consiste em encontrar uma forma de comunicação para que seja possível efetuar o contato com um Chaveiro Automotivo, para a abertura do veículo, nesse caso tem de se levar em consideração o tempo que será gasto esperando o profissional e o valor que o mesmo cobrará pelo serviço prestado, dependendo do modelo e do fabricante do automóvel que pode onerar o custo;
- A terceira consiste em instalação do dispositivo no interior do veículo, anteriormente ao fato ocorrido, pois não existe uma data prevista para esse problema ocorrer, com isso existe a prevenção para o problema descrito acima.

Nesse item é levado também em consideração o Smartphone, que necessita estar previamente carregado para a realização da operação. Se o veículo possuir o dispositivo e o problema vier a ocorrer, fica fácil solucionar, é preciso apenas utilizar o Smartphone e selecionar a opção desejada no aplicativo para que se possa abrir o vidro ou destravar a porta e ter acesso ao interior do veículo.

1.4. Escopo do Trabalho

O projeto consiste em implementação dos seguintes itens em caráter primário: Travas Elétricas e Vidros Elétricos.

O protótipo poderá ter o acréscimo de mais uma funcionalidade, como o Controle de Locomoção, que consiste no aperfeiçoamento das funcionalidades do projeto. Tal funcionalidade representa o funcionamento básico de um controle automatizado de uma suspensão a ar, ou seja, não é escopo do projeto desenvolver um controle remoto para um veículo.

O projeto consiste em desenvolver mais uma funcionalidade para o Smartphone com Android, por meio da aplicação Amarino e da comunicação *Bluetooth* que interliga celular e o dispositivo Arduino, que está acoplado ao protótipo e que servirá como interpretador de funcionalidades oriundas do aplicativo.

O protótipo é um simulador automotivo que contém as funcionalidades que representam de forma simples os itens descritos acima, onde o circuito eletrônico (dispositivo Arduino e módulo *Bluetooth*) representará a central de conforto automotiva, estruturas com os servomotores (Travas e Vidros Elétricos), sinais luminosos e sonoros como retorno da operação realizada, em caso de sucesso ou falha. O Smartphone será utilizado com a aplicação Amarino e por meio deste é que as funções serão ativadas de acordo com a necessidade do usuário.

1.5. Resultados Esperados

O produto gerado (microcontrolador conectado ao circuito eletrônico que representa a central de conforto do veículo, protótipo, programação e aplicativo para o celular) consiste em mais uma forma de utilização do celular assim como das próprias funcionalidades do veículo alvo da simulação, podendo ser utilizada em casos extremos, um bom exemplo seria o caso de se esquecer à chave no interior do veículo e fechar os vidros e travar as portas, com a utilização da solução proposta seria possível utilizar o aplicativo combinado com o *hardware* para destrancar o carro ou abrir algum dos vidros para que seja possível ter acesso ao interior

do veículo. O projeto (produto final) não é de âmbito comercial e o mesmo será um projeto acadêmico (a programação do *software* para o celular e o microcontrolador não serão vendidos, pelo fato de não ser uma solução de baixo custo, por causa do custo muito elevado para a aquisição dos dispositivos utilizados no decorrer do projeto).

O resultado esperado do projeto é que o mesmo possa ter sucesso no decorrer da implementação, e que as tarefas selecionadas por meio do aplicativo no Smartphone possam ocorrer com sucesso, como a ativação do vidro e da trava elétrica, assim como também do controle de locomoção que representa o funcionamento básico de uma suspensão a ar. Outro item importante é a integração das plataformas Arduino e Android, que são os itens essências para o desenvolvimento do projeto.

1.6. Estrutura do Trabalho

A estrutura da monografia foi definida em seis capítulos, sendo que cada um contribui de forma única para a descrição das diversas teorias necessárias, etapas de desenvolvimento e conclusões acerca do projeto, conforme é descrito a seguir:

- Capítulo 1 – Introdução – descrição inicial do problema a ser resolvido.
- Capítulo 2 – Apresentação do Problema – descrição do contexto do problema, levando em consideração as diversas soluções dadas aos problemas, e definição de como essa proposta aqui mencionada pode solucionar o problema descrito no item 1.1.
- Capítulo 3 – Referencial Teórico e Tecnológico – nesse capítulo é apresentado o referencial teórico e tecnológico que reforça a teoria necessária para compreensão de funcionamento dos itens abordados no decorrer do projeto. Neste capítulo serão abordados os temas acerca das plataformas Arduino e Android, Amarino, Servomotores, *Bluetooth* e o funcionamento de Vidros e Travas Elétricas Automotivas.
- Capítulo 4 – Desenvolvimento do Projeto – nesse capítulo são apresentados os métodos utilizados para o desenvolvimento da solução, e descreve de forma sucinta a visão do projeto conglomerando os itens para o bom andamento do projeto. Ainda nesse capítulo são descritas as especificações acerca dos diversos *hardwares*, *software*s que foram necessários para a construção, implementação do projeto.
- Capítulo 5 – Aplicação do Projeto – nesse capítulo são descritos os problemas que sugiram no decorrer da implementação e de que forma impactaram na solução

final. As soluções foram geradas a partir dos testes realizados e com isso a necessidade de melhorar o funcionamento do protótipo.

- Capítulo 6 – Conclusão – o último capítulo desse projeto diz respeito às conclusões e aos problemas enfrentados no decorrer do desenvolvimento do protótipo. Ainda nesse capítulo, serão descritas algumas propostas para projetos futuros com relação ao projeto.

2. APRESENTAÇÃO DO PROBLEMA

O capítulo dois consiste na descrição de dois problemas e como cada um pode ser solucionado com a utilização da solução proposta.

2.1. Contexto Geral do Problema

Com o avanço da tecnologia a chave do veículo que inicialmente era utilizada somente para abrir ou fechar as portas ou dar partida no mesmo recebeu novas funcionalidades, como aumento da segurança por meio da utilização de codificação e alarme para a realização de outras funções de forma integrada, como controle de travas e vidros.

Apesar desse tipo de tecnologia estar em constante evolução e cada vez mais sofisticada, é necessário levar em consideração dois problemas que podem ocorrer e que é alvo de solução no decorrer desse projeto:

- **Problema Um: Travar o veículo com a chave dentro:** um fator relativamente comum levando em consideração a falta de tempo e a correria do dia-a-dia, sendo que os alarmes atuais possuem um *timer* na sua programação que efetua o travamento das portas e vidros após um determinado tempo, e o local onde ocorreu esse problema é afastado da cidade e de difícil acesso a rede de telefonia.
- **Problema Dois: Molhar a chave do veículo e pelo fato da maçaneta da porta não ter orifício e não ser possível destravar as portas do automóvel:** esse tipo de eventualidade pode ocorrer em festas ou até mesmo em casa, e com isso ocorrer o dano no equipamento interno e/ou no momento em que for acionada, a mesma não responder de forma correta.

Com o problema número um descrito acima, pode surgir duas soluções para o referido problema. A primeira consiste em encontrar um meio de comunicação que funcione, para que um chaveiro profissional possa ser acionado e solicitado para a abertura do veículo. Já a segunda é necessário saber da viabilidade de buscar a chave reserva, e assim como na primeira hipótese é necessário entrar em contato com um familiar para solicitar que o mesmo traga o dispositivo reserva para realizar a abertura do automóvel, caso a mesma exista.

Já no problema número dois, a solução poderia ser a mesma descrita para o problema um, ou seja, contatar um chaveiro profissional para que o mesmo possa consertar o equipamento danificado, ou efetuar a troca do dispositivo dependendo da situação do mesmo, com isso o montante gasto para essa solução tem um valor de custo muito elevado.

Levando em consideração os problemas iniciais e as três soluções propostas anteriormente é que esse projeto está baseado, na qual uma solução móvel pode ser utilizada

para realizar o destravamento dos dispositivos com a utilização do aplicativo *Amarino* instalado no Smartphone com Android e com a utilização do *Bluetooth*, lembrando do alcance do dispositivo.

Com base nesse problema é que será possível solucionar o problema proposto da seguinte maneira, que consiste em conectar o Arduino ao *Bluetooth* e com o auxílio do Smartphone com Android e Aplicação Amarino definir qual função será ativada, das quais os vidros ou travas elétricas poderão ser utilizados para ter acesso ao interior do veículo.

Para que a solução represente o objetivo real da implementação é necessário que a solução proposta esteja integrada ao veículo, e caso aconteça esse tipo de problema é que a mesma poderá ser utilizada.

2.2. Tecnologias Existentes

O veículo da fabricante Ford, modelo Fusion possui um teclado numérico localizado acima da maçaneta da porta do motorista, e tem por objetivo dar acesso caso a senha digitada seja a mesma cadastrada previamente. Com isso, é possível utilizar o mesmo sem ser necessário ter em mãos a chave do próprio veículo ou a mesma tenha sido deixada em seu interior, conforme é visto na Figura 2.1.

Figura 2.1 - Teclado Numérico na porta do Ford Fusion.

Fonte – (http://www.guiaveiculos.net/noticias.php?cd=125)

Na referida solução acima, o objetivo dessa funcionalidade é muito interessante, pois caso o proprietário não queira utilizar a chave do veículo durante uma caminhada, por exemplo, é possível que o mesmo possa deixar a chave no seu interior, e caso seja necessário abrir o veículo é preciso digitar a senha para abertura do mesmo. Já o ponto negativo desse tipo de sistema é acerca do efeito visual gerado, onde não é muito interessante ter no exterior do veículo um teclado numérico e que somente uma pequena parte do setor automotivo possui esse tipo de sistema.

Um ponto positivo da solução móvel alvo desse projeto é que a sua utilização permite controlar os vidros e travas por meio de um celular, que é um dispositivo muito difundido e com uma gama extensa de utilizações e cada vez mais esse tipo de tecnologia tem por missão o conforto e praticidade. Já o ponto negativo é que a tecnologia utilizada para a construção do protótipo possui um valor relativamente elevado se comparado a produção de uma única unidade.

A idéia desse projeto é permitir que a solução gerada seja utilizada por uma gama mais ampla de veículos e não somente pelo veículo referenciado anteriormente. O celular por ser um dispositivo amplamente difundido e com diversas utilizações e sabendo-se que cada vez mais pessoas não vivem sem o mesmo é que o produto final pode ser utilizado para realizar as funções de vidros e travas.

3. BASES METODOLÓGICAS PARA RESOLUÇÃO DO PROBLEMA

No capitulo três ocorre à descrição das plataformas do Arduino e Android e dos itens utilizados para desenvolver o protótipo do projeto. Neste capítulo é descrita a teoria necessária para compreensão de funcionamento e como cada uma encaixa no contexto da solução.

3.1. Microcontrolador

Os microcontroladores são dispositivos semicondutores e possuem uma vasta utilização no mercado, seja na área de automação residencial (microondas), automação industrial ou embarcada (robôs da linha de montagem automotiva ou computadores de bordo) e até automação predial (elevadores e alarmes). Por ser limitado, geralmente não é empregado caso seja necessário utilizar uma quantidade de memória acima de sua capacidade [GIMENEZ, 2002].

O microcontrolador é composto por um processador, memória e pinos de I/O (ou E/S – entrada/saída) e existe uma variedade finita de modelos à venda no mercado para os mais variados tipos de implementação e capacidade. Sendo que os principais itens levados em questão são: memória interna (capacidade), tempo de processamento (velocidade) e quantidade de pinos de I/O e interrupções.

Para utilizar os microcontroladores é necessário conhecer algumas informações básicas acerco do seu funcionamento, conforme está descrito abaixo:

- **Bits:** abreviação de dígito binário (*binary* digit), que corresponde aos valores zero (0) ou um (1) lógico. Zero lógico geralmente corresponde a zero volt (V) e um lógico, a 5 ou 3 volts (V).
- **Bytes:** representação numérica composta de oito (8) bits. Pode representar números de 00h (= 0_{10}) a FFh (= 255_{10}). É principalmente empregada na representação das instruções que o microcontrolador é capaz de executar [GIMENEZ, 2002].
- **Registradores:** são equivalentes a uma RAM, só que interna à CPU. Servem para armazenamento temporário de informações de utilidade interna ou externa à CPU.
- **Microprocessador ou Unidade Central de Processamento (CPU):** é a pastilha do microprocessador, o "cérebro" do sistema, e tem competência para acionar e se comunicar com todas as vias (controle de endereços, memórias e I/O), só que sempre seguindo ou obedecendo às diretivas gravadas na ROM.
- **Instruções:** define uma única ação (tarefa) que o microcontrolador pode executar por vez. As ações das instruções podem corresponder a operações de leitura e escrita de

conteúdos dos registradores ou nas posições de memórias ou a operações lógicas e aritméticas [GIMENEZ, 2002].

- **Interrupções:** são pinos de acesso externo que permitem interromper o microprocessador, que então interrompe suas tarefas atuais e atende aquelas planejadas pela interrupção solicitada [NICOLOSI, 2004].
- **Programa ou *Software*:** definido por um *conjunto de instruções* arranjadas de maneira organizada por um programador (profissional especialista em computação), com o objetivo de informar ao microcomputador o que deve ser executado, ou seja, o programa informa ao microcomputador quais as tarefas que devem ser executadas ao longo do tempo. O programa sempre deve estar armazenado em uma unidade de armazenamento de informações, ou seja, em uma memória.
- **Firmware:** programa (*software*) que está exclusivamente armazenado em uma memória não-volátil (ROM/PROM/EPROM/EEPROM) de um dispositivo microcontrolado. No mínimo, o firmware tem a finalidade principal de programar a forma de operação do *hardware* (número de portas de entrada e saída, forma de comunicação serial, forma de operações dos *timers*/contadores) e de definir suas condições iniciais de operação.
- ***Hardware*:** são as partes eletrônicas que compõem um microcomputador, como por exemplo, uma placa mãe [GIMENEZ, 2002]. Efetuando uma analogia com o projeto em questão, seria como descrever cada dispositivo utilizado no decorrer da implementação, como é o caso do Arduino, *Bluetooth*, Protoboard e Servomotores.
- **Tipos de Memória:** as memórias são responsáveis pelo armazenamento dos dados e estão subdivididas em dois grupos e conseqüentemente em vários tipos:
    - **Voláteis:** perdem o conteúdo quando a alimentação é desligada.
        - **SRAM:** memória de acesso aleatório estática.
        - **FRAM:** memória de acesso aleatório ferromagnética.
        - **DRAM:** memória de acesso aleatório dinâmica.
        - **DDR-RAM**: memória de acesso aleatório digital dupla.
    - **Não voláteis:** mantém o conteúdo mesmo sem a alimentação.
        - **ROM:** memória de apenas leitura.
        - **PROM:** memória de apenas leitura programável.
        - **EPROM:** memória de apenas leitura programável e apagável (por luz ultravioleta).

- **EEPROM:** memória de apenas leitura programável e apagável eletronicamente.
- **Flash:** similar à EEPROM [GUIMARÃES, 2007].

3.1.1. Microcontrolador *ATmega328P*

O microcontrolador *ATmega328P* é utilizado nos dispositivos Arduino Uno, Arduino 2009 e Arduino *Bluetooth* ou BT. É um microcontrolador de 8 bits e o chip é fabricado pela empresa ATMEL. A arquitetura do equipamento é a RISC (*Reduced Instruction Set Computer* ou Computador com um Conjunto Reduzido de Instruções – representa uma linha de processadores que apresentam um conjunto simples e pequeno de instruções e que utilizam a mesma quantidade de tempo para executar as mesmas) e a freqüência de operação é cerca de 16 MHz. O referido dispositivo ainda possui memórias do tipo Flash, EEPROM e RAM, sendo que cada uma delas possui 32KB, 1KB e 2KB, respectivamente. Na sua composição ainda estão vinte e três entradas e saídas digitais, com tensão de operação de varia de 1.8 a 5.5V e baixo consumo de energia [ATMEL, 2010].

Na Figura 3.1, está à representação do microcontrolador utilizado no decorrer desse projeto.

Figura 3.1 - Microcontrolador *ATmega328P*.

Fonte – (*http://multilogica-shop.com/atmega328*)

3.2. Arduino

O Arduino é uma plataforma de prototipagem eletrônica open-source, e é baseada na flexibilidade do *hardware* e na facilidade de uso por meio de *software*. A sua destinação é para qualquer tipo de pessoa que esteja interessada em desenvolver, criar projetos ou ambientes interativos [BANZI, 2008].

Por ter uma grande quantidade de portas de entradas e saídas, proporciona uma extensa gama de aplicações, que pode ser na utilização conjunta de sensores, motores, dentre outros dispositivos para a geração de diversas funcionalidades.

A linguagem de programação da Plataforma Arduino é o *Wiring* e o ambiente de desenvolvimento é baseado em *Processing*, e dependendo do projeto a ser desenvolvido podem ser realizados de duas formas:

- ***Stand-alone:*** os sensores conectados as portas de entrada ou saída enviam os dados para o microcontrolador, que por sua vez realiza o processamento e posteriormente ocorre a ativação de alguma tarefa.
- ***Dependente de Software:*** o Arduino está conectado tanto ao computador quanto aos sensores. Após leitura de dados (analógico ou digital) e envio para um *software* em execução (por exemplo, Flash, Visual Studio, Java) é que serão devidamente processados e utilizados da forma mais conveniente. Os dados são enviados com a utilização da porta serial.

O diferencial da plataforma Arduino é que os produtos podem ser adquiridos ou montados seguindo as referências dos componentes e nas descrições das placas. A licença open-source do Arduino permite a modificação de alguma funcionalidade ou outra função dependendo somente da necessidade da mesma.

Na Figura 3.2, estão representadas todas as placas que serão utilizadas no decorrer do projeto, conforme a necessidade.

Figura 3.2 - Placas da Plataforma Arduino.

Fonte – (*O Autor*)

3.2.1. Arduino Duemilanove

A placa utilizada para o desenvolvimento do projeto é o Arduino Duemilanove ou “2009”, sendo baseada no microcontrolador *ATmega328P*. A mesma possui catorze portas digitais das quais seis podem ser utilizadas como saídas PWM, ainda possuem seis portas analógicas que dão suporte a maioria dos projetos caseiros, hobby ou acadêmicos.

O Arduino Duemilanove possui conexão USB (que pode ser utilizado para fornecimento de energia, programação do dispositivo ou conexão do *hardware* com o computador, conforme descrito anteriormente), conector de energia, conector ICSP e botão de reset para ser utilizado caso algum problema esteja ocorrendo. Para utilizar o Arduino 2009 é necessário inicialmente conectá-lo a porta USB ou fonte de energia externa.

Uma curiosidade acerca do nome ‘*Duemilanove*’ é que ele significa 2009 em italiano, que representa o ano da data de lançamento e a placa faz parte de uma série de Controladores Arduino USB. O Arduino UNO é a atualização do dispositivo e não ocorram modificações de grande impacto, conforme pode ser visto da Figura 3.3.

Figura 3.3 - Placas Arduino Duemilanove (frente e trás) e Arduino UNO.

Fonte – (*O Autor*)

A tensão de operação da placa pode ser uma fonte externa de 6 a 20 volts, caso o fornecimento de energia seja menor que 7V, o pino de 5V irá fornecer um valor menor do que a tensão do mesmo e ocorrer a instabilidade. Caso a fonte de energia forneça mais de 12V, o regulador de tensão pode superaquecer e danificar a placa. É recomendado utilizar de 7 a 12V para a alimentação do dispositivo.

A alimentação do Arduino Duemilanove pode ser com a utilização dos pinos de V_{in}, 5V, 3.3V e GND (terra), conector Jack ou USB, conforme está representado na Figura 3.4.

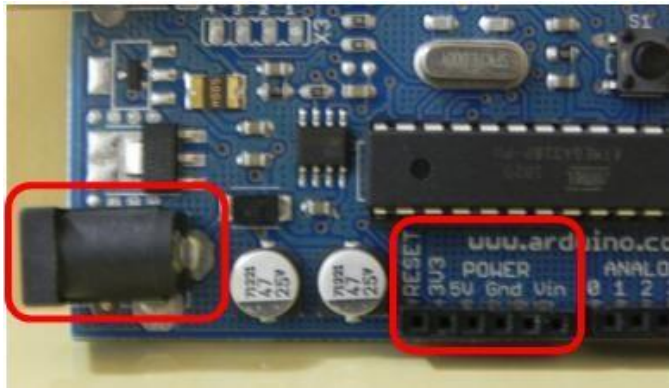


Figura 3.4 - Alimentação do Arduino Duemilanove.

Fonte – (*O Autor*)

O microcontrolador *ATmega328P* possui 32KB de memória flash para armazenamento de código, sendo que 2KB são utilizados pelo *bootloader*, que é o gerenciador de inicialização do dispositivo. No microcontrolador ainda existem 2KB de SRAM e 1KB de memória EEPROM.

Dependendo da alimentação utilizada, os pinos de entrada e saída podem fornecer ou receber no máximo 40mA e 50mA, para as tensões de 5V e 3.3V, respectivamente.

3.2.2. Arduino UNO

O Arduino UNO segue a mesma linha de dispositivos USB, e consiste na atualização do Arduino Duemilanove, mas a diferença é que não utiliza o chip FTDI para controle de USB-Serial e sim o *ATmega8U2*.

Assim como ocorre no Arduino Duemilanove, o nome do Arduino UNO está escrito em italiano e representa o lançamento da versão Arduino 1.0, que é a versão base para os dispositivos desta linha de produtos, e na Figura 3.5 está a sua representação visual da mesma.

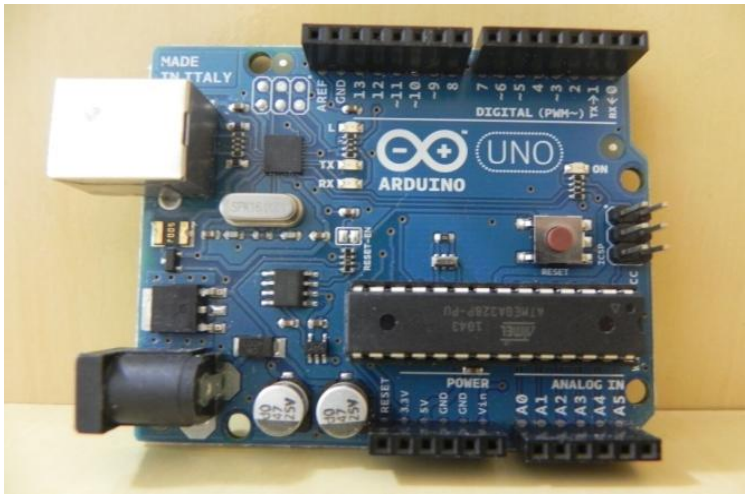


Figura 3.5 - Placa Arduino UNO.

Fonte – (*O Autor*)

3.2.3. Arduino *Bluetooth*

A placa Arduino *Bluetooth* ou Arduino BT é mais um dispositivo desta plataforma de desenvolvimento open-source e também está baseada no microcontrolador *ATmega328P*, mas diferentemente das outras citadas anteriormente esse modelo possui algumas características que dificultam sua utilização. Por utilizar à conexão *Bluetooth*, a mesma não possui USB e é necessário ter um pouco de atenção quanto à forma de fornecimento de energia para a placa, ainda assim tudo que é necessário para dar suporte ao microcontrolador e também para a programação remota, mas a comunicação serial não é compatível com *headset Bluetooth* ou outros dispositivos de áudio.

O Arduino *Bluetooth* está integrado ao módulo Bluegiga WT11, que está especialmente configurado para esse dispositivo e fornece comunicação para computadores, telefone e outros dispositivos com a tecnologia *Bluetooth*. Para se comunicar, o módulo WT11 compartilha via serial a comunicação com o microcontrolador *ATmega328P* utilizando os pinos RX e TX da placa. Por padrão o dispositivo está configurado por 115200 bps para transmissão de dados. A energia pode ser conectada sem o auxilio de *plug*, conector ICSP e botão de reset caso alguma coisa esteja gerando algum erro ou a mesma com algum problema, que não é resetada automaticamente, se uma nova programação for ser gravada é necessário resetar manualmente e no tempo exato, caso contrário gerará um erro, muito comum nesse tipo de placa, conforme é visto na Figura 3.6.

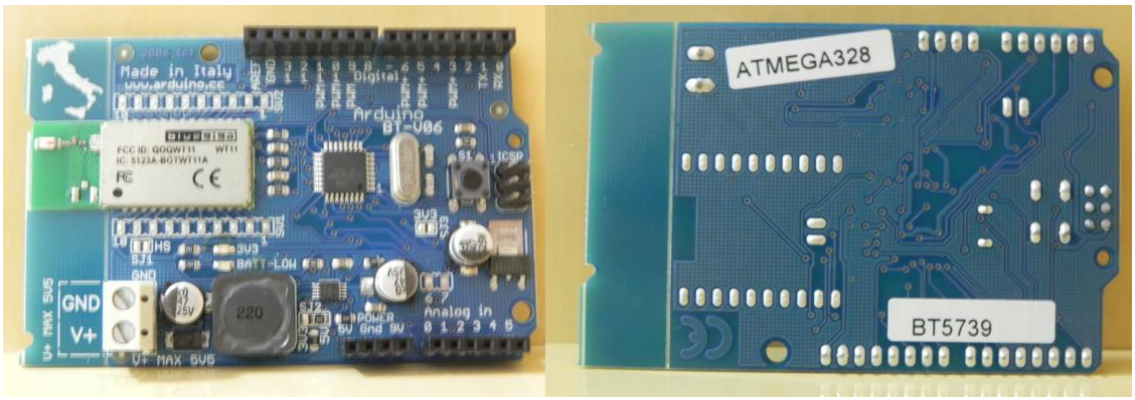


Figura 3.6 - Placas Arduino *Bluetooth* (frente e trás).

Fonte – (*O Autor*)

Assim como no Arduino UNO e Duemilanove, as funções do microcontrolador *ATmega328P* são as mesmas, ora descrita anteriormente, no item.

A alimentação conectada através de parafusos conforme representado na Figura 3.7 pode variar de 1.2V a 5.5V no máximo, caso uma tensão mais elevada ou polaridade invertida for adicionada, pode ocorrer dano ou destruição da mesma. Na placa ainda existe a impressão da tensão de 9V, que não deve ser utilizada por danificar a placa.

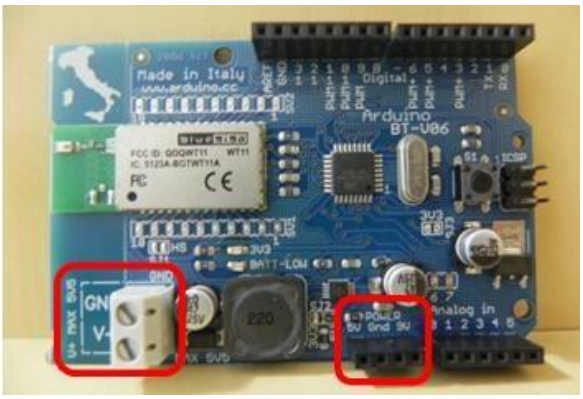


Figura 3.7 - Alimentação do Arduino *Bluetooth*.

Fonte - (*O Autor*)

3.2.4. Aplicação Arduino Development Enviroment

Para o desenvolvimento da programação para a plataforma Arduino é necessário utilizar o ambiente desenvolvido em *software* livre, e por meio deste é possível escrever a programação, gravação no *hardware* e realizar o debug do código para as placas Arduino. No ambiente de desenvolvimento é necessário utilizar a linguagem de programação *Wiring*, que é uma variação de outras linguagens, como o C e o C++. Algumas bibliotecas específicas já fazem parte do aplicativo e por se tratar de *software* livre, existe a possibilidade de adicionar ou criar outras, dependendo do que será criado ou modificado. O Arduino IDE possui a capacidade de reconhecimento de todas as estruturas da Linguagem C e também de alguns recursos da Linguagem C++, com o código criado e compilado é possível realizar o '*upload*' para a placa Arduino, ou seja, caso o código obedeça às regras da linguagem e nenhum erro seja identificado, é gerado o código binário equivalente ao que foi criado e então é gravado no microcontrolador, para posterior utilização. Na Figura 3.8 está parte do código desenvolvido para o protótipo.

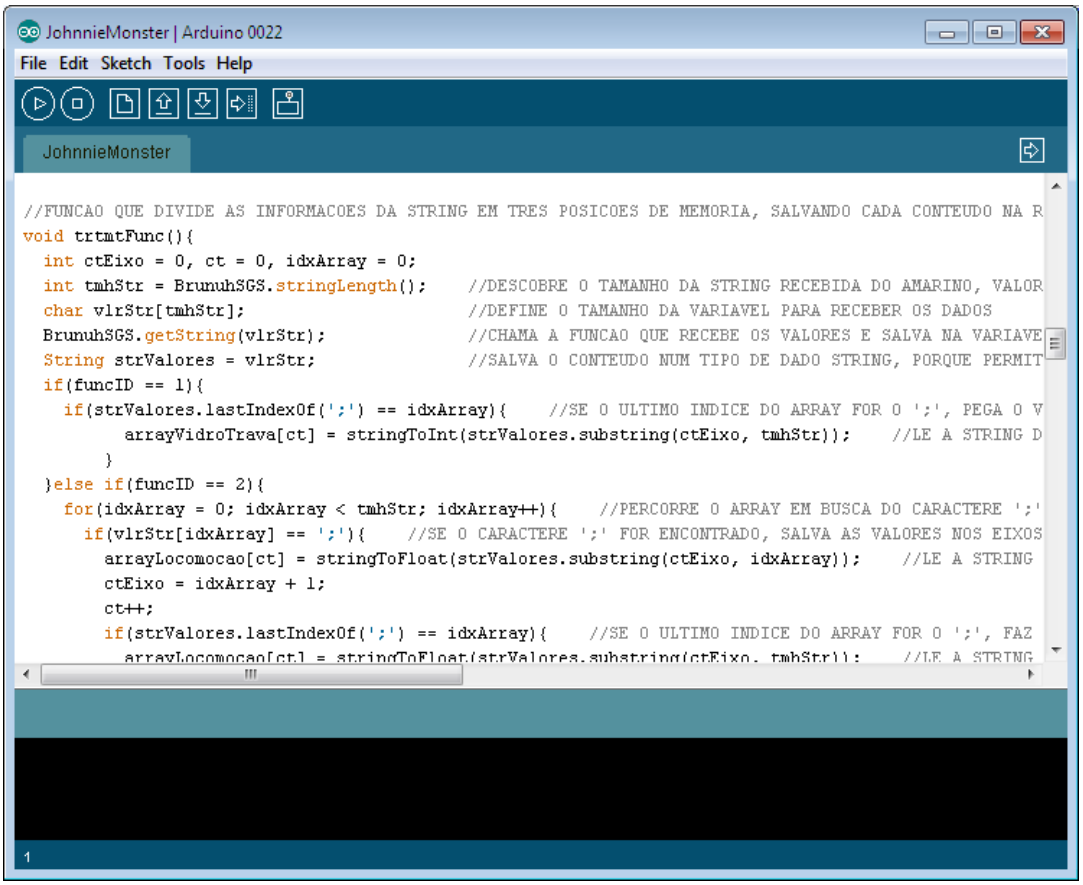


Figura 3.8 - Ambiente de Desenvolvimento Arduino – Parte do Código do Protótipo.

Fonte – (*O Autor*)

3.3. *Bluetooth*

A tecnologia de rede pessoal sem fio *Bluetooth* surgiu a partir de uma necessidade de conexão de telefones móveis, PDA's, notebook's e acessórios pessoais sem a utilização de cabos. No ano de 1994, a empresa L. M. Ericsson teve interesse em conectar seus dispositivos móveis sem a real necessidade de utilização dos cabos e para isso uniu-se a outras quatro empresas (IBM, Intel, Nokia, Toshiba) para formar um consórcio SIG (Special Interest Group) que tinha um interesse em comum, desenvolver um padrão sem fio para interconectar dispositivos de computação e comunicação e ainda acessórios, utilizando rádios de curto alcance, baixa potência e baixo custo. O então projeto *Bluetooth*, assim denominado foi em homenagem ao rei viking Harald Blaatand (*Bluetooth*) II (949-981) por ter conquistado a Dinamarca e a Noruega, por não ter utilizado cabos para o seguinte feito [TANEMBAUM, 2003]. E da mesma forma que o rei teria conseguido unificar suas tribos, o padrão *Bluetooth* procurar unir diferentes tecnologias e uma curiosidade acerca do logotipo (Figura 3.9) é que o símbolo é a união de algumas runas (conjunto de alfabetos relacionados que utilizam letras características) nórdicas, como no caso de: ᚼ - H (Hagall) e ᛒ - B (Berkanan), que são as

letras iniciais do rei viking e o “Dente-Azul” na sua tradução literal deram origem ao símbolo comumente utilizado, o *Bluetooth*.

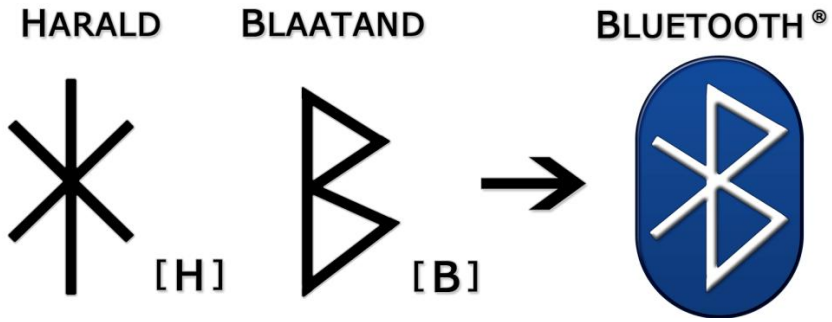


Figura 3.9 - Origem do Logotipo do *Bluetooth*.

Fonte – (*O Autor*)

O Consórcio SIG emitiu uma especificação de 1.500 páginas da versão 1.0 no ano de 1999 e este documento foi utilizado pelo IEEE para examinar as redes especiais sem fio e assim sucessivamente efetuar sua padronização, levando em consideração os detalhes técnicos e algumas modificações para o padrão, dando origem ao 802.15.1 – *Bluetooth*. Os padrões do SIG e IEEE são um pouco divergentes, espera-se que ambas possam convergir para um único padrão a fim de aprimorar ainda mais esse tipo de tecnologia.

3.3.1. Arquitetura do *Bluetooth*

A arquitetura do *Bluetooth* é definida por dois tipos de redes: *piconets* e *scatternet*.

- ***Piconet***: consiste em um nó mestre e até sete nós escravos ativos desde que situados dentro de uma distância de 10 metros. Embora uma *piconet* possa ter um máximo de sete escravos é possível adicionar um oitavo sendo que este último ficará no estado estacionado (*parked state*), e isso significa que o escravo desse estado fica sincronizado com o mestre, mas não pode tomar parte na comunicação até que seu estado seja modificado. Já que somente oito estações podem estar ativas num *piconet*, retirar uma estação do estado estacionada significa levar uma das estações ativas para o estado estacionado [FOROUZAN, 2006]. O item (*a*) da Figura 3.10 demonstra uma rede *piconet*.
- ***Scatternet:*** a combinação de redes *piconets* pode formar uma *scatternet*. Uma estação escrava numa *piconet* pode se tornar a mestre em outra *piconet*. Com isso esta estação pode receber mensagens do mestre da primeira rede (como escrava) e agir como mestre na qual repassaria às escravas da segunda *piconet*. Uma estação pode pertencer simultaneamente a duas redes *piconets*. No item (*b*) está a representação desse tipo de rede.

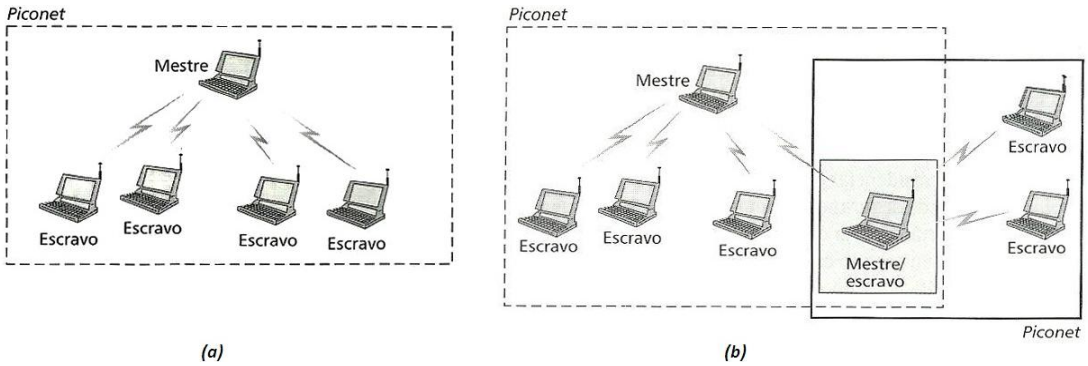


Figura 3.10 - Redes *Piconets* e *Scatternets.*

Fonte - (*FOROUZAN, 2006*).

3.3.2. Pilha de Protocolos do *Bluetooth*

O padrão *Bluetooth* tem muitos protocolos agrupados livremente em camadas. A estrutura de camadas não segue o modelo OSI, o modelo TCP/IP, o modelo 802 ou qualquer outro modelo conhecido. Porém, o IEEE está trabalhando na modificação do *Bluetooth* para que ocorra a melhor adaptação ao modelo 802. A arquitetura básica de protocolos do *Bluetooth*, modificada pelo comitê 802, está representada na Figura 3.11.

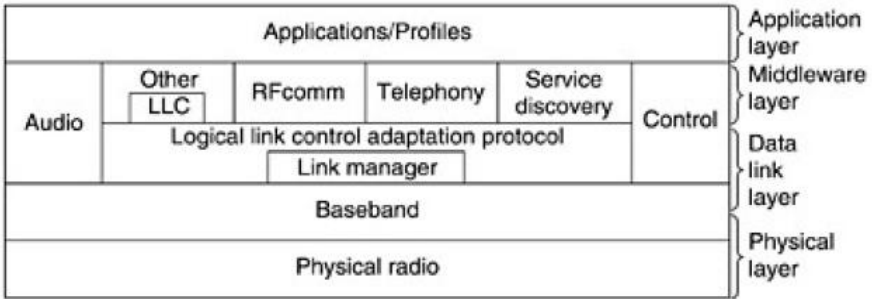


Figura 3.11 - Arquitetura de Protocolos do *Bluetooth*.

Fonte - (*TANEMBAUM, 2003*).

A camada inferior é a camada física de rádio, que corresponde muito bem à camada física nos modelos OSI e 802. Nessa camada ocorre a transmissão e a modulação de rádio. A camada de banda base é de certa forma an*álog*a a subcamada MAC, mas também inclui elementos da camada física. É função dessa camada é gerenciar como o mestre controla os *slots* de tempo e como esses *slots* são agrupados em quadros.

Em seguida, temos uma camada com um grupo de protocolos até certo ponto inter-relacionados. O gerenciador de enlaces cuida do estabelecimento de canais lógicos entre dispositivos, incluindo o gerenciamento de energia, autenticação e qualidade de serviço. O protocolo de adaptação de controle de enlace lógico (freqüentemente chamado *L2CAP*) isola as camadas superiores dos detalhes de transmissão. A subcamada LLC é análoga ao padrão

802, mas é tecnicamente diferente. Como seus nomes sugerem, os protocolos de áudio e controle lidam respectivamente com o áudio e o controle.

A próxima camada é a camada *middleware*, que contém uma mistura de diferentes protocolos. O LLC do 802 foi inserido aqui pelo IEEE para manter a compatibilidade com as outras redes 802. Os protocolos *RFcomm*, de telefonia e de descobertas de serviços são originais. O protocolo *RFcomm* (comunicação por freqüência de rádio) é o protocolo que emula a porta serial padrão encontrada nos computadores pessoais para conectar o teclado, o mouse e o modem, entre outros dispositivos. Foi projetado para permitir que dispositivos de tecnologia antiga o utilizem com facilidade. O protocolo de telefonia é um protocolo de tempo real utilizado pelos três perfis orientados para voz. Que também gerencia a configuração e o encerramento de chamadas. Por fim, o protocolo de descoberta de serviços é usado para localizar serviços na rede.

As aplicações e os perfis se localizam na camada superior. Eles utilizam os protocolos das camadas inferiores para cumprir suas funções. Cada aplicação tem seu próprio subconjunto dedicado de protocolos. Dispositivos específicos, como um fone de ouvido em geral somente possui os protocolos exigidos por essa aplicação e nenhum outro [TANEMBAUM, 2003].

3.3.3. Camada de Rádio do *Bluetooth*

Na camada de rádio a função principal é transferir os bits do dispositivo mestre para o escravo ou vice-versa. Por ser um sistema de baixa potência o seu alcance é limitado a 10 metros e a banda de operação é a ISM de 2,4 GHz. A subdivisão da banda consiste em 79 canais de 1 MHz para cada um. A modulação do canal é dado pelo chaveamento por deslocamento de freqüência, com 1 bit por Hz, fornecendo assim uma taxa de dados bruta igual a 1 Mbps, sendo que grande parte desse espectro é consumido por overhead. Para ocorrer à alocação dos canais de maneira uniforme, é utilizado o espectro de dispersão de saltos de freqüência com 1600 hops/s e um tempo de parada de 625 s. Todos os nós em uma piconet saltam simultaneamente, com o mestre ditando a seqüência de saltos [TANEMBAUM, 2003]

3.3.4. Camada de Banda Base do *Bluetooth*

A camada de banda base é a estrutura mais próxima de uma subcamada MAC que o *Bluetooth* possui. Com isso, ocorre a transformação do fluxo bruto de bits em quadros e dessa forma a definição de alguns formatos importantes. De modo simples, o mestre de cada *piconet*

define uma série de *slots* (fatias) de tempo de 625 s, sendo que as transmissões do mestre ocorrem nos *slots* pares e as transmissões dos escravos nos *slots* ímpares. Esse tipo de transmissão é tradicional multiplexação por divisão de tempo, onde o mestre possui metade dos *slots* e os escravos compartilham a outra metade. Os quadros podem ter 1, 3 ou 5 *slots* de duração.

Cada quadro é transmitido sobre um canal lógico, chamado enlace (link), entre o mestre e um escravo. Existem dois tipos de enlaces. O primeiro enlace é o ACL (*Asynchronous Connection Link* – link de conexão assíncrona), que é utilizado para dados em comutação de pacotes e é disponível por intervalos irregulares. A origem dos dados é da camada L2CAP do lado da transmissão e são entregues na camada L2CAP no lado da recepção. O tráfego ACL é entregue em uma base de *melhor esforço* e com isso não é dada nenhuma garantia de que a informação será recebida. Os quadros podem ser perdidos e pode ser necessário retransmitir em determinados momentos onde ocorrer à perda. Um escravo somente pode ter um enlace ACL para seu mestre [FOROUZAN, 2006].

O segundo enlace é o SCO (*Synchronous Connection Oriented* — conexão orientada por sincronia), e é utilizada para tráfego de dados de tempo real, como as conexões telefônicas. A esse tipo de canal é alocado um *slot* fixo em cada sentido. Devido à natureza crítica dos enlaces SCO, os quadros enviados sobre eles nunca são retransmitidos. Em vez disso, pode ser usada a correção de erros antecipada para proporcionar alta confiabilidade. Um escravo pode ter até três enlaces SCO com seu mestre. Cada enlace SCO pode transmitir um canal de áudio PCM de 64.000 bps [TANEMBAUM, 2003].

3.3.5. A camada L2CAP do *Bluetooth*

A camada L2CAP possui três funcionalidades importantes. A primeira consiste em aceitação de pacotes de até 64 KB das camadas superiores e os divide em quadros para transmissão. Na outra extremidade, os quadros são montados novamente em pacotes. Em segundo lugar, ocorre a utilização de multiplexação e a demultiplexação de várias origens de pacotes. Quando um pacote é novamente montado, a L2CAP determina a qual protocolo da camada superior ocorrerá à entrega.

Em terceiro lugar, a camada L2CAP lida com os requisitos de qualidade de serviço, tanto quando os enlaces são estabelecidos quanto durante a operação normal. Também e negociado em tempo de configuração o tamanho máximo de carga útil permitido, a fim de impedir que um dispositivo de pacotes grandes afogue um dispositivo de pacotes pequenos.

Esse recurso é necessário, porque nem todos os dispositivos podem manipular o pacote máximo de 64 KB.

3.3.6. Módulo BlueSMiRF Gold

O módulo BlueSMiRF da *SparkFun Eletronics* é o mais recente dispositivo com tecnologia *Bluetooth* para transmissão de dados sem fio e funcionam como modem para recepção e transmissão de dados seriais, RX e TX respectivamente. O fluxo de dados varia desde 9600 a 115200 bps podendo ser transmitido de um computador para o destino ou de um dispositivo móvel, como por exemplo, um Smartphone. Seguindo as orientações da *SparkFun*, o módulo consegue alcançar até 106m quando utilizado ao ar livre, graças a antena instalada internamente e a utilização da Classe 1 do *Bluetooth*, que permite maiores alcances de distâncias.

A alimentação do módulo suporta tensões de 3.3 até 6V. O dispositivo não pode ser conectado diretamente a uma porta serial porque pode ocorrer um dano e para o mesmo é necessário um adaptador de USB - Serial, caso exista a real necessidade de modificação de nome, senhas e outras configurações. Na Figura 3.12 está a ilustração do módulo.

Figura 3.12 - Módulo *Bluetooth* - BlueSMiRF Gold.

Fonte – (*O Autor*).

Por ser um *hardware* muito pequeno, foi otimizado para que a transmissão ocorra com total integridade e que não ocorra perda dos dados durante a realização do mesmo, possui também um baixo consumo de energia, sendo que uma média de consumo gira em torno de 25 mA. Pelo regime de trabalho ser com salto de freqüências, o módulo consegue operar em ambiente com dispositivos de RF como *Wi-Fi*, *Zigbee* sem perder a eficiência. A criptografia dos dados permite total segurança e confidencialidade.

3.4. Android

O mercado de celulares está crescendo cada vez mais. Estudos mostram que hoje em dia mais de três bilhões de pessoas possuem um aparelho celular, e isso corresponde a mais ou menos metade da população mundial. E hoje em dia os usuários comuns estão procurando

cada vez mais celulares com diversos recursos como câmeras, músicas, *Bluetooth*, ótima interface visual, jogos, GPS, acesso a internet e emails e TV Digital [LECHETA, 2010].

Com essa crescente evolução da mobilidade, os mercados corporativos viram a necessidade de incorporar aplicações em dispositivos móveis para agilizar seus negócios e também integrar os diversos sistemas existentes.

As diversas aplicações que desempenham variadas funcionalidades nos celulares podem estar conectadas e sempre realizar sincronia das informações para verificação da real identidade da pessoa que está ativando um serviço, por exemplo. Com isso, a praticidade gerada por esse tipo de serviço é algo que está causando grandes revoluções no mundo tecnológico, e nos dias atuais temos diversos bancos oferecendo serviços das mais variadas funcionalidades, a fim de aperfeiçoar processos e minimizar custos, onde um simples Smartphone pode ser utilizado para verificação de saldos bancários ou pagamentos de faturas, algo tão surpreendente que em muitos países mais desenvolvidos permitem o pagamento de mercadorias com a utilização do celular, como se fosse um cartão de crédito. Para que toda essa realidade possa atender a todos é necessário que uma plataforma flexível e confiável esteja apita a tornar tudo isso mais viável [LECHETA, 2010].

Na Figura 3.13 é apresentado o logotipo do Sistema Operacional Android.

Figura 3.13 - Logotipo do Android.

Fonte - (www.android.com).

O Android é a resposta do Google para ocupar esse espaço. Consiste em uma nova plataforma de desenvolvimento para aplicativos móveis, baseada em um sistema operacional Linux, com diversas aplicações já instaladas e, ainda, um ambiente de desenvolvimento bastante poderoso, ousado e flexível. Outra boa notícia é que pode utilizar a consagrada linguagem Java para desenvolver as aplicações, usufruindo de todos os recursos provenientes da tecnologia, na Figura 3.14 está à representação visual do sistema operacional Android.

Figura 3.14 - Tela do Android no Smartphone Samsung Galaxy S.

Fonte – (*O Autor*).

O Android causou um grande impacto quando foi anunciado, atraindo a atenção de muita gente. Podemos dizer que isso aconteceu porque por trás dele está nada mais nada menos do que o Google, a empresa que está revolucionando a internet. Entretanto, não é apenas o Google que está na jogada, e sim um grupo formado por empresas lideres do mercado de telefonia como a Motorola, LG, Samsung, Sony Ericsson e muitas outras. Esse grupo, chamado de Open Handset Alliance (OHA) foi criado com a intenção de padronizar uma plataforma de código aberto e livre de celulares, justamente para atender a todas as expectativas e tendências do mercado atual [LECHETA, 2010].

O fato de o Android ser de código aberto contribui muito para o seu aperfeiçoamento, uma vez que desenvolvedores de todos os lugares do mundo podem contribuir para seu código-fonte, adicionando novas funcionalidades ou simplesmente corrigindo falhas.

É possível integrar aplicações de uma forma simples, sejam elas desenvolvidas por usuários comuns, sejam aplicações nativas. Por exemplo, imagine que sua aplicação precise consultar a agenda de contatos para selecionar determinado amigo, e logo depois visualizar o endereço dele em um mapa. Bom, que existe a agenda de contatos e o Google Maps no Android muitos já devem ter ouvido falar ou até mesmo utilizado, mas integrar aplicações é uma das palavras-chaves em aplicações corporativas, e a arquitetura do Android foi criada justamente pensando nisso.

Outro ponto forte do Android é que seu sistema operacional é baseado em Linux, e o mesmo se encarrega de gerenciar a memória e os processos. Isso permite que diversas aplicações possam ser executadas ao mesmo tempo, permitindo que aplicações em segundo

plano consigam executar sem que o usuário perceba, enquanto que ele está acessando a internet ou atendendo uma ligação [LECHETA, 2010].

3.4.1. Aplicativo Amarino

Com a constante evolução dos Smartphones e cada um com uma nova funcionalidade, seja por meios de sensores ou de utilitários do próprio dispositivo é que foi desenvolvida uma aplicação que permite criar uma infinidade de projetos que visam à capacitação de pessoas para "externalizar" eventos provenientes do telefone e utilizá-los das mais diversas formas que a criatividade consiga proporcionar, ou seja, transformar os dados do mundo virtual em algo "palpável" como a ativação de uma lâmpada ou controle de um robô, por exemplo.

O Amarino é uma ferramenta que permite conectar um dispositivo com sistema operacional Android aos microcontroladores Arduino utilizando a conexão *Bluetooth* e com isso geração de diversas funcionalidades através da utilização dos eventos oriundos do celular.

Pelo fato dos celulares permitirem a personalização de diversos itens e representarem em muitos casos o dono do dispositivo, é que o projeto Amarino foi baseado. Um grupo de tecnologia do Instituto de Tecnologia de Massachusetts – MIT iniciou o projeto com o intuito de promover a utilização dos eventos dos Smartphones para criação de soluções personalizadas, sem a real necessidade de experiência em programação.

Na Figura 3.15, pode ser visto o aplicativo Amarino em funcionamento no Smartphone Samsung Galaxy S.

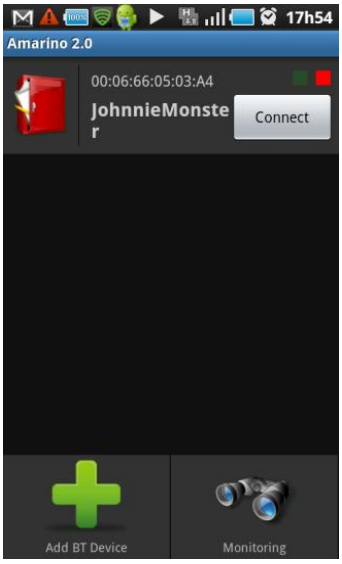


Figura 3.15 - Tela inicial do Aplicativo Amarino do Smartphone.

Fonte – (*O Autor*).

Com o aplicativo é possível utilizar os diversos recursos dos celulares para envio e recebimento de informações. No desenvolvimento do protótipo, o Amarino foi utilizado para

envio dos dados oriundos do acelerômetro, com isso é possível controlar outro dispositivo utilizando a praticidade e inovação ocasionada pela seguinte união, no caso Android, Arduino e Amarino. O aplicativo possui diversas funcionalidades, conforme é representado a Figura 3.16, como Bússola Digital, Sensor de Proximidade, Sensor de Magnetismo, Envio de SMS e tantos outros. E a escolha do acelerômetro foi ocasionada porque em diversos testes realizados comprovaram o melhor funcionamento entre os demais e em caso de ajuste poderia ser realizado com maior facilidade.

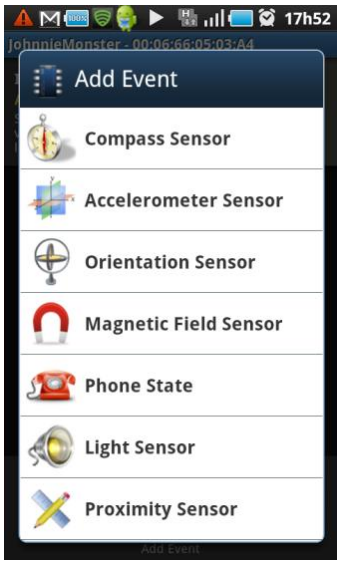


Figura 3.16 - Tela com funcionalidades do Amarino.

Fonte – (*O Autor*).

O Amarino possui um *log* que permite verificar o que está acontecendo no momento, ou seja, é possível saber se a conexão com o dispositivo foi realizada com sucesso ou algum erro não permitiu realizar tal tarefa, também é possível verificar qual a informação enviada ou recebida, com isso é possível saber qual o valor que um determinado sensor está enviando e qual a função que o outro dispositivo realizou, conforme está demonstrado na Figura 3.17.

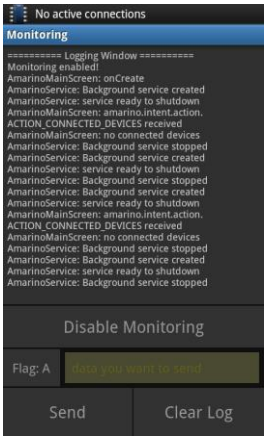


Figura 3.17 - Tela de *Log* do Amarino.

Fonte – (*O Autor*).

3.5. Servomotor

Os servomotores são dispositivos que possuem uma infinidade de aplicações seja no âmbito da robótica (comumente utilização em pequenos projetos e simulações de robôs) ou até mesmo para a utilização nos diversos tipos de indústrias para automação de mecanismos (como portas, brinquedos etc.) das mais variadas funcionalidades. São largamente utilizados por sua precisão e facilidade, seja em movimentos de rotação ou linear ou até mesmo para posicionamento em determinado ponto. Os servomotores recebem sinais elétricos e convertem os mesmos em movimentos de rotação ou deslocamentos lineares, por existir variados modelos e diferentes tipos de motores e engrenagens, pode ser que para determinados modelos haja um processo de realimentação para que o servomotor possa realizar a operação com precisão. Na Figura 3.18 são apresentados os modelos utilizados no protótipo do projeto.

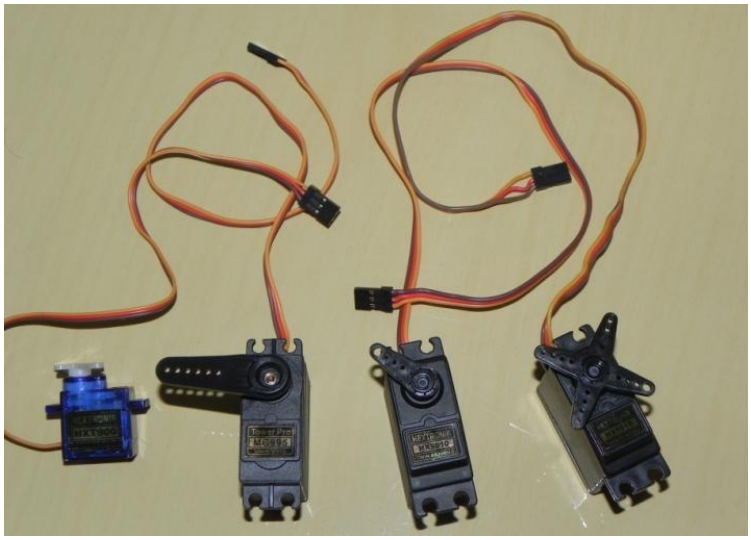


Figura 3.18 - Servomotores utilizados para os testes iniciais.

Fonte – (*O Autor*).

3.5.1. Composição dos Servomotores

Os servomotores são compostos genericamente por motores de pequeno porte, circuitos eletrônicos para controle das funcionalidades, potenciômetro que está conectado ao eixo (define a movimentação da engrenagem, com “início” e “fim” do percurso), engrenagens para redução e cabeamento de energia/terra/dados.

Conforme está descrito em [FRANCISCO, 2008], a função do pequeno motor encontrado no interior do servo é girar um conjunto de engrenagens para amplificar ou diminuir a velocidade de rotação do eixo, onde estão interligados os diversos dispositivos que são alvos desse estudo. Com o processo de movimentação das engrenagens é possível obter força para movimentação e assim conseguir realizar determinada tarefa, seja abrir ou fechar um vidro, travar ou destravar o pino da porta ou ainda mesmo locomover o protótipo para um

estado diferente do inicial. Conforme as informações são recebidas pelo servomotor, as mesmas devem ser processadas e com isso realizar o devido posicionamento do eixo, ou seja, movimentando-o para conseguir mudar um objeto interligado ao eixo do lado esquerdo para o direito, ou vice-versa por exemplo. No mercado possuem diversos tipos e variadas configurações de servomotores e alguns são capazes de movimentar objetos mais pesados com um simples movimento do eixo, fazendo com que esses dispositivos produzam muita força levando em consideração seu tamanho. A Figura 3.19, é apresentada uma abordagem geral dos itens mencionados anteriormente.

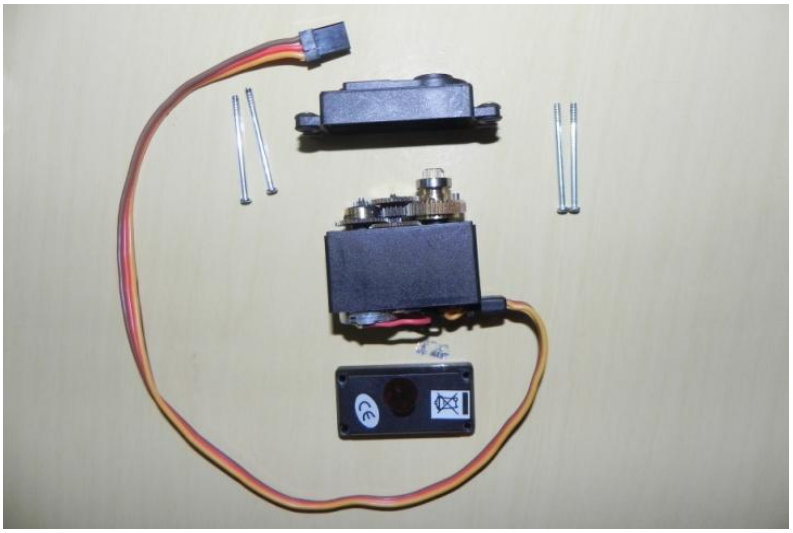


Figura 3.19 - Itens de um servomotor.

Fonte – (*O Autor*).

3.5.2. Descrição do Funcionamento de um Servomotor

No interior da servomotor, existem dois itens essenciais para o funcionamento do dispositivo, que é o circuito eletrônico que determina a funcionalidade e o potenciômetro, onde a união faz com que o sistema tenha um *feedback*, ou seja, haja uma realimentação para saber em que posição o eixo está localizado. Com isso é possível efetuar um controle instantâneo e determinar qual a angulação que o eixo se movimentará para realização da operação, no caso o objetivo de sua utilização.

O pequeno motor é interligado por uma série de engrenagens até o eixo, ou seja, a cada ‘X’ voltas percorridas pelo motor existe uma correspondência muito menor, com esse ajusto de ângulo a ângulo é possível determinar com precisão a posição que se deseja movimentar ou aumento de torque para conseguir proporcionar maior velocidade ao eixo e para que tudo isso ocorra de forma correta sem nenhum equívoco é que o potenciômetro está preso a estrutura do servomotor (caixa plástica) e ao eixo. O valor da resistência determina a ação do motor/eixo, efetuando um controle de monitoração contínuo e preciso.

Caso o valor da resistência do potenciômetro esteja diferente do que o circuito eletrônico está aguardando, um impulso é enviado para correção da diferença de posicionamento. Com isso o circuito consegue controlar a todo instante a posição angula do eixo do servo, caso o mesmo esteja na angulação equivocada o motor é acionado para que a devida correção seja efetuada, caso contraria nenhuma reação é desencadeada. Um fato interessante é que a tensão aplicada ao motor é proporcional a distância de movimentação, se existir a necessidade de percorrer um curto espaço têm-se a limitação da velocidade e se for necessário girar grandes percursos é que o mesmo obtém a velocidade máxima para que a correção seja efetuada em um curto espaço de tempo. Tendo essa propriedade a denominação de controle proporcional. Em princípio o servomotor permite uma movimentação de 0 a 180 graus [FRANCISCO, 2008].

3.5.3. Controle do Ângulo de Rotação de um Servomotor

Para ocorrer a rotação no motor dos servomotores é necessário que o impulso (tempo *ON* – tempo que o servomotor estará ativo) receba um comando na entrada, ativando um sinal PWM (*Pulse Width Modulation* – Modulação por Largura de Impulso) e ocorra a movimentação do eixo. Esse sinal consiste em um comprimento de onda na qual ocorre a variação da duração do tempo T_{on}, com o período sendo mantido com valor fixo.

Por existirem diversos tipos de servomotores, a largura mínima e máxima do impulso depende única e exclusivamente do seu modelo. Em geral, os servomotores trabalham de forma bem parecida uns dos outros e recebem na sua entrada impulsos com a duração de:

- *Impulso de 1 ms*: o eixo gira no sentido anti-horário até que atinja o limite do intervalo de rotação, nesse caso corresponde ao ângulo de 0º;
- *Impulso de 1,5ms*: o eixo gira até o momento de estabilidade, ou seja, o centro do intervalo de rotação, correspondendo ao o ângulo de 90º;
- *Impulso de 2 ms*: o eixo gira no sentido horário até que atinja o outro limite do intervalo de rotação, correspondendo a 180º (dependendo do modelo é que existe a possibilidade de um valor menor ou um pouco maior que o de 180º).

Em termos simples, têm-se que impulsos entre 1 e 1,5 ms proporcionam posições intermediárias de 0 a 90º e de 1,5 a 2 ms possuem posições dentro do intervalo de 90 a 180º, conforme descrição da Figura 3.20.

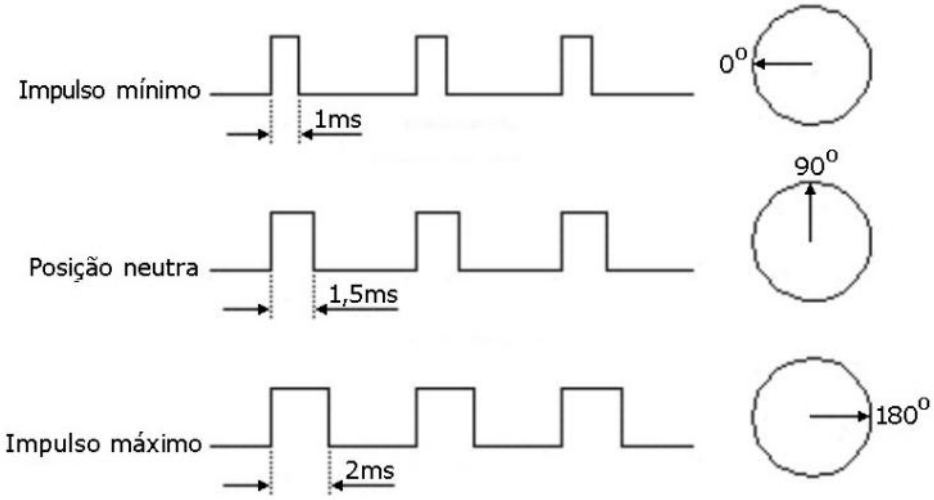


Figura 3.20 - Diagrama de Controle do Ângulo de Rotação dos Servomotores.

Fonte – (*FRANCISCO, 2008*).

3.5.4. Modificação do Servomotor para Simulação de Sistema de Locomoção

Para que fosse possível utilizar os servomotores para a simulação do sistema de locomoção, foi necessário modificar o conjunto circuito eletrônico e potenciômetro, que é o realimentador do sistema e fornece um *feedback* informando em qual posição o eixo do sistema se encontra.

O potenciômetro está conectado ao circuito eletrônico e ao eixo, com isso é possível descobrir com precisão sua localização com base no ângulo. Na Figura 3.21 são ilustrados os itens internos de um servomotor (motor, circuito eletrônico e potenciômetro).

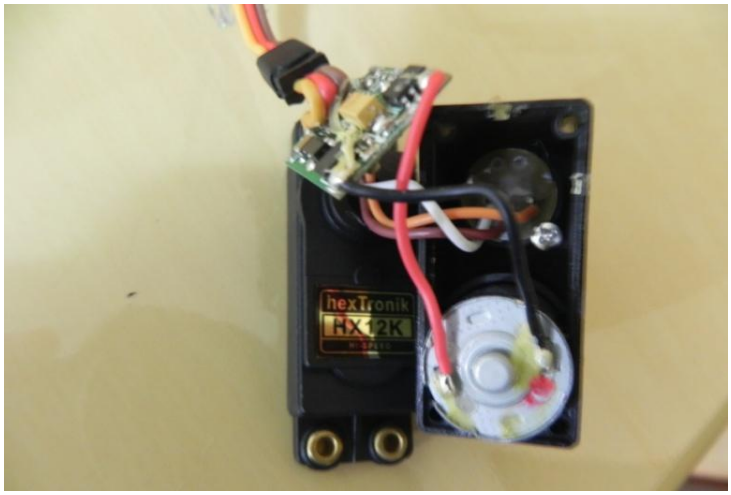


Figura 3.21 - Interior de um Servomotor.

Fonte – (*O Autor*).

Para realizar a modificação foi necessário realizar a dessoldagem do potenciômetro e no soldar no seu lugar dois resistores de 2.2KΩ conectados em paralelo, que funciona como divisor de tensão e dessa forma infere ao servo que o potenciômetro está no meio do curso

original, ou seja, levando em consideração que um servomotor funciona de 0º a 180º, todas às vezes o sistema receberá como resposta que ele está em 90º. Na Figura 3.22 está a representação da modificação descrita anteriormente.

Figura 3.22 - Substituição do Potenciômetro por Resistores em Paralelo no Servomotor.

Fonte – (*O Autor*).

Após a realização da substituição do potenciômetro pelos resistores em paralelo é necessário cortar a trava que impede o eixo girar mais de 180º, para realizar esse passo foi necessário retirar a tampa da caixa superior e com isso as engrenagens do sistema estarão à mostra, conforme é representado no item (a) da Figura 3.23. A maior engrenagem é a do eixo e após a sua retirada somente é necessário cortar o limitador, conforme é descrito no item (b). Após a conectar novamente a engrenagem modificada ao sistema e remontar a tampa inferior e superior, foi possível concluir o processo de modificação. Testes comprovam o novo funcionamento do servomotor e com isso a simulação da locomoção será possível. A função de *feedback* será realizada por meio de programação e com isso será possível controlar melhor a funcionalidade dependendo somente da regra a ser utilizada.

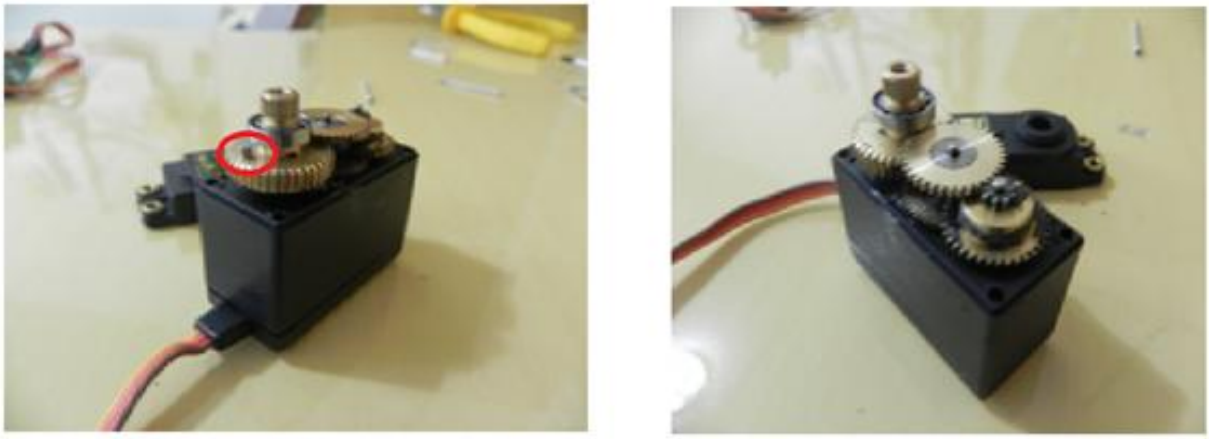


Figura 3.23 - Remoção do Limitador das Engrenagens no Servomotor.

Fonte – (*O Autor*).

3.5.5. Utilização do Servomotor para Simulação de Vidros e Travas

A escolha dos servomotores para a composição do projeto e simulação das ativas desempenhadas pelos motores de vidros e travas elétricos não foi por acaso, e sim por conhecer o poder de sua precisão e nas diversas aplicabilidades que poderiam ocorrer no protótipo durante a implementação do projeto. Também foi levado em consideração o batente (trava da engrenagem do eixo) que limita a movimentação do eixo, justamente porque nos vidros e travas elétricas veiculares é necessário saber onde inicia e finaliza, para que não ocorre nenhum dano ao dispositivo em geral caso o servomotor tente movimentar mais que sua capacidade. No protótipo foram utilizados os servomotores da marca Hextronik e Tower Pro, modelos HXT5010, HX12K e MG955, com torques de 6.9 kg/cm, 10 kg/cm e 15 kg/cm, respectivamente. Conforme está descrito na Figura 3.24.

Figura 3.24 - Servomotores Utilizados na Simulação de Vidros e Travas (localizados ao centro) e Locomoção do Protótipo (conectados as rodas traseiras).

Fonte – (*O Autor*).

3.6. Sistemas Embarcados

Os Sistemas Embarcados Automotivos englobam uma grande quantidade de conteúdo e nesse sentido somente serão tratados os assuntos de Vidros e Travas Elétricas, que fazem parte do Sistema Trio Elétrico.

O sistema trio elétrico é um conjunto formado pelos opcionais: alarme, travas elétricas e levantador elétrico dos vidros, e tem como objetivo principal aumentar o conforto do motorista e dos passageiros [GUIMARÃES, 2007].

Esses opcionais são intensamente procurados no mercado por questões de segurança veicular, por exemplo, o alarme e também por gerar segurança para os objetos deixados no

interior do veículo, já nos casos de vidros e travas elétricas são geralmente utilizados por questões de conforto e comodidade.

Conforme é representado na Figura 3.25, vê-se como é o comportamento da maioria dos segmentos de automóveis no Brasil e sua grande maioria a procura para pelo Sistema Trio Elétrico. Comparando esses dados com o de outros países europeus é notável a diferença de consumo, justamente porque os índices de furtos são muito baixos, com isso esses itens não são muito solicitados, apesar de sempre solicitarem as travas elétricas, pela questão do conforto e vidros elétricos somente nas portas dianteiras [GUIMARÃES].

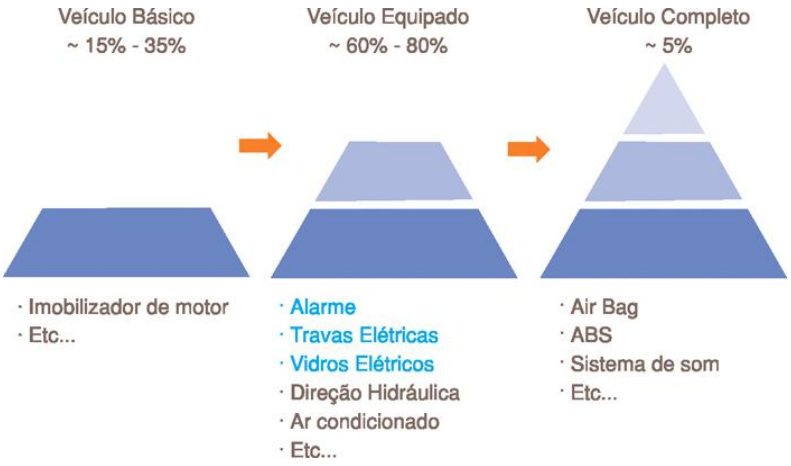


Figura 3.25 - Comportamento médio do mercado.

Fonte – (*GUIMARÃES, 2007*).

3.6.1. Travas Elétricas

As travas elétricas são módulos eletromecânicos formados por pequenos motores elétricos e algumas engrenagens são instaladas nas portas do veículo, e são responsáveis pelo destravamento, travamento e, eventualmente, pelo travamento mecânico (*deadlock*) das portas [GUIMARÃES, 2007].

Nesse contexto, são apresentados três conceitos acerca das formas de travamento das portas:

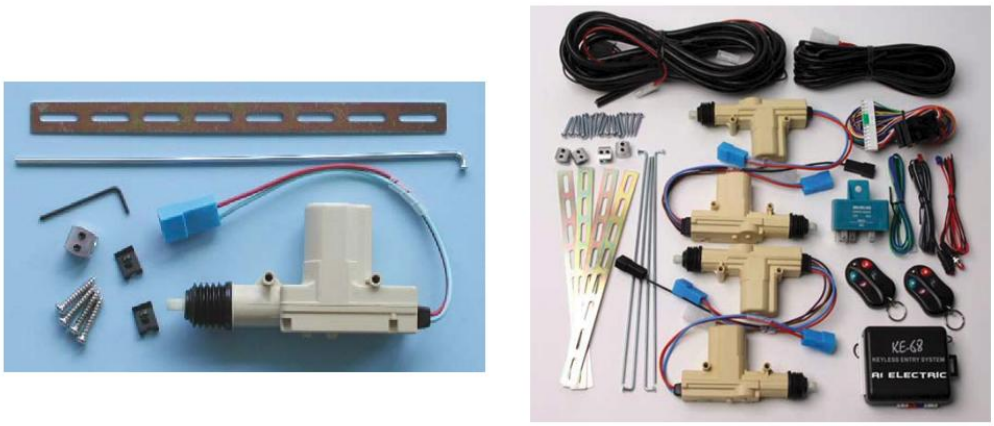
- **Destravar – *unlock***: significa liberar mecanicamente as portas para que elas possam ser efetivamente abertas com uso de maçanetas internas ou externas.
- **Travar – *lock***: significa travar as portas, impedindo a abertura pelas maçanetas externas. As maçanetas internas podem ser utilizadas em condições específicas, variáveis conforme o veículo.
- **Travamento mecânico – *deadlock ou deadbolt***: representa uma proteção adicional ao veículo e ao seu interior. Com o deadlock acionado, as portas não

podem ser abertas pelas maçanetas externas nem internas. Esse sistema está relacionado à segurança do veículo e não ao conforto propriamente dito. Além disso, não é encontrado em todos os sistemas de travas elétricas disponíveis no mercado [GUIMARÃES, 2007].

O módulo de travas é o responsável pelo controle das fechaduras elétricas e além das fechaduras das portas, existem outros três atuadores no sistema de travas:

- **Atuador da portinhola de combustível:** esse item tem por função a deliberação de acesso ao bocal de abastecimento de combustível.
- **Atuador da tampa traseira:** determina o acesso ao porta-malas.
- **Atuador de destranca do porta-malas:** realiza o destravamento da tampa traseira seja ela por interruptor da funcionalidade no interior do veículo ou por acionamento via controle remoto da chave.

Na Figura 3.26 estão representados alguns tipos de sistemas referenciados anteriormente. No item (a) está o conjunto de peças de um sistema de destranca do porta-malas, no item (b) está o kit básico de um sistema de travas elétricas.



(a) (b)

Figura 3.26 - Conjunto de Peças dos Sistemas de Travas Automotivas.

Fonte – (*GUIMARÃES, 2007*).

As funcionalidades variam desde modelo a montadoras, cada uma com um item a mais de segurança que são oferecidos para os consumidores e dependendo do sistema utilizado o item conforto dá lugar a segurança dos ocupantes. Em alguns modelos é possível notar a utilização de sensores de colisão (*crash sensor*), que faz a detecção de colisão e determina ao módulo eletrônico de controle para realizar o destravamento das portas para facilitar a saída do veículo ou a remoção de acidentados em casos mais graves.

3.6.2. Vidros Elétricos

Os vidros elétricos são sistemas eletromecânicos formados pelos motores elétricos e algumas alavancas e/ou cabos de aço são instalados nas portas do veículo, ficando responsáveis pela abertura ou pelo fechamento dos vidros. São as chamadas máquinas do vidro elétrico [GUIMARÃES, 2007].

Alguns termos utilizados e funções desempenhadas por esse tipo de sistema serão descritos a seguir.

- **Fechamento automático – *comfort closing*:** função que fecha todas as janelas no momento em que o veículo é travado. Esse tipo de função necessita das travas elétricas para funcionar.
- **Subida e descida expressas – *express up and express down*:** possibilitam a subida e a descida das janelas com apenas um toque no interruptor de comando.
- **Proteção antiesmagamento – *pitch protection*:** reverte o sentido do deslocamento das janelas quando elas forem fechadas e algo obstruir seu caminho, como uma mão, por exemplo. Evita acidentes que poderiam ser fatais em alguns casos.
- **Alívio interno de pressão – *internal pressure relief*:** toda vez que uma das portas é aberta, uma das janelas também é aberta automaticamente, em alguns casos de centímetros, voltando a se fechar imediatamente após o fechamento da porta. O intuito dessa função é eliminar a sensação de pressão nos ouvidos ao fechar as portas com todas as janelas fechadas [GUIMARÃES, 2007].

O sistema de vidros elétricos funciona com a utilização de três itens basicamente: máquinas de vidro elétrico (funcionalidade de subir e descer o vidro), interruptores (determinam qual função realizar) e módulo eletrônico de controle que controla o sistema como um todo. Na Figura 3.27, são representados três tipos de modelos comumente utilizados nesse tipo de sistema. No item (a) é representado o Conceito Tesoura (*Scissors*).
O item (b) mostra o Conceito Cabo de Aço (*Bowden Cable*). Já o item (c) apresenta o Conceito Cabo de Aço (*Goldie Cable*).

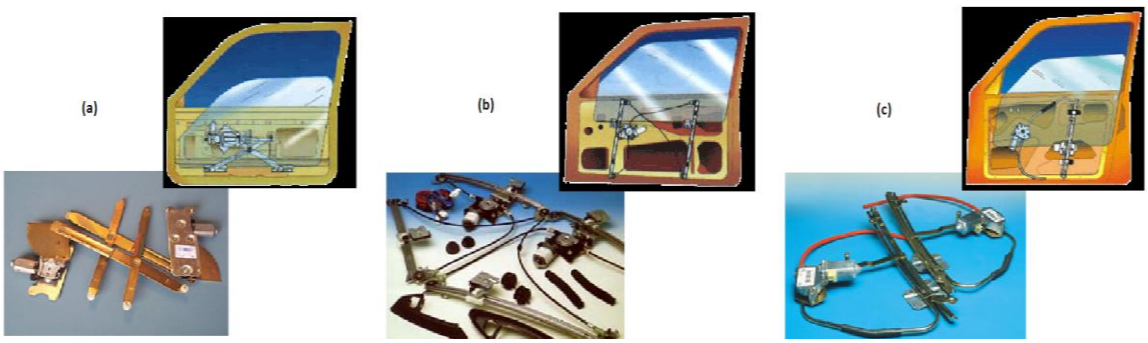


Figura 3.27 - Modelos de Sistemas de Vidros Elétricos Automotivos.

Fonte – (*GUIMARÃES, 2007*).

Os interruptores dos vidros elétricos estão separados em dois grupos com conceitos mecânicos diferentes: os *push-push* e os *push-pull* [GUIMARÃES, 2007].

- **Empurra-empurra – *Push-push*:** para subir ou descer os vidros, o motorista ou passageiro precisa aperta os interruptores. Esse tipo de interruptor deve ser montado em superfícies bem inclinadas. Se montado em superfícies pouco inclinadas ou paralelas ao solo, ele pode causar acidentes, especialmente em crianças e animais, pois ambos costumam apoiar no interruptor de subida da janela e podem prender parte do corpo entre a janela e o quadro da porta. Em superfícies bem inclinadas, essa possibilidade é extremamente minimizada.
- **Empurra-puxa – *Push-pull*:** para subir a janela, o interruptor deve ser puxado. Para descer a janela, o interruptor precisa ser pressionado. Além de mais seguro, esse sistema é o mais intuitivo à operação e mais atraente do ponto de vista do design. É, atualmente, uma tendência mundial [GUIMARÃES, 2007].

Na Figura 3.28 estão representados os dois tipos de interruptores mencionados anteriormente, o *item (a)* representa o Empurra-empurra – *Push-push* e o *item (b)* representa o Empurra-puxa – *Push-pull*.

Figura 3.28 - Modelos de Interruptores de Vidros Elétricos Automotivos.

Fonte – (*GUIMARÃES, 2007*).

4. MODELO PROPOSTO

4.1. Apresentação do Modelo Proposto.

O modelo proposto tem por objetivo gerar uma solução que consiste em abrir ou fechar os vidros e travar e destravar as travas da porta utilizando um aplicativo instalado no Smartphone com Android conectado ao *hardware* por meio do *Bluetooth*. No presente projeto é necessária a utilização de diversos componentes tanto de *software* e *hardware* de variadas tecnologias e com a utilização em conjunto é que será gerada a solução proposta, conforme breve descritivo citado anteriormente.

O funcionamento dos dispositivos ocorre da seguinte forma: com o aplicativo Amarino instalado no Smartphone com Android é necessário efetuar uma conexão com o *hardware* que está acoplado ao protótipo, ou seja, o Amarino efetua um pareamento com o *Bluetooth* conectado ao Arduino e com isso é possível enviar e receber dados. *Log*o após a fase de conectado é possível utilizar o acelerômetro do dispositivo celular para realização de tarefas previamente determinadas, dependendo de como está o posicionamento do Smartphone nos eixos X, Y e Z é que uma possível tarefa de simulação de vidro ou trava poderá ocorrer.

Na Figura 4.1 segue um resumo em forma de diagrama das etapas de funcionamento do protótipo.

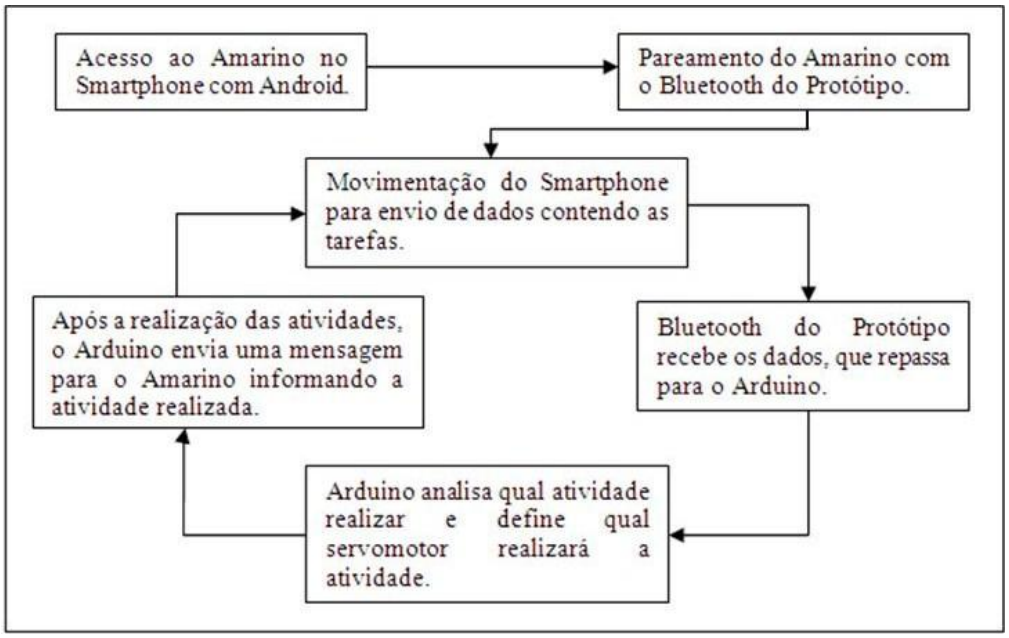


Figura 4.1 - Diagrama de Funcionamento do Modelo Proposto.

Fonte - (*O Autor*).

O dispositivo Arduino é conectado ao Smartphone por meio de uma conexão *Bluetooth*, sendo gerenciada através do aplicativo Amarino e conforme ocorre à

movimentação do celular é que o *software* realizará o envio dos dados para que o *hardware* identifique a atividade e informe sucessivamente o que foi realizado, ou seja, abrir ou fechar os vidros ou travar e destravar as travas. Na Figura 4.2 segue um breve esquemático de como estão conectados os *hardwares* e *softwares* para a solução proposta.

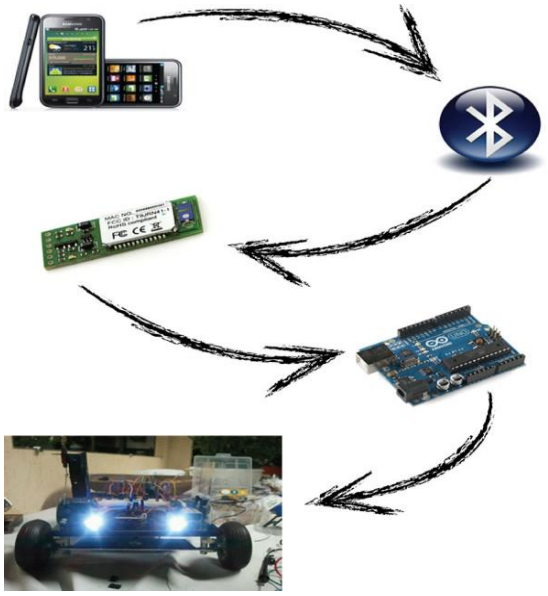


Figura 4.2 - Esquemático do Projeto Físico.

Fonte - (*O Autor*).

4.2. Descrição das Etapas do Modelo

O projeto é caracterizado pela realização de diversas etapas que possuem uma meta em comum, que é gerar a solução proposta. Durante a implementação, etapas como montagem, programação do microcontrolador, testes e modificações são necessárias para que seja possível construir um protótipo que represente de forma simples a solução real e com isso reunir em um pequeno espaço todos os componentes utilizados do projeto que realização as funções.

4.3. Descrição da Implementação

A seguir serão descritas as diversas etapas do projeto como um todo, afim de que fique claro todas as modificações realizadas durante a implementação do protótipo e como cada uma contribuiu para a solução final.

4.3.1. Configuração dos Servomotores com Arduino

Com os servomotores adquiridos para a simulação das atividades foi necessário testar cada modelo em separado para entender o funcionamento e definir o modo de operação referente às atividades que cada uma representará.

Inicialmente os servomotores do projeto seriam os modelos HXT900 e HXT5010 da fabricante Hextronik, pelo fator custo ser relativamente baixo, mas após testes de funcionamento e levando em consideração o peso do protótipo os equipamentos referenciados anteriormente foram utilizados para testes de exaustão.

Com os modelos referenciados anteriormente foram realizados os seguintes testes: definição do início e fim da locomoção do eixo.

O primeiro teste consistia em descobrir o ponto inicial do eixo, ou seja, o ponto que representa o ângulo de 0º. Na Figura 4.3 estão os modelos utilizados para testes, com os pontos iniciais (*a*) e com isso seria necessário ajustar o braço conectado ao eixo para que o mesmo represente o ângulo desejado (*b)*.

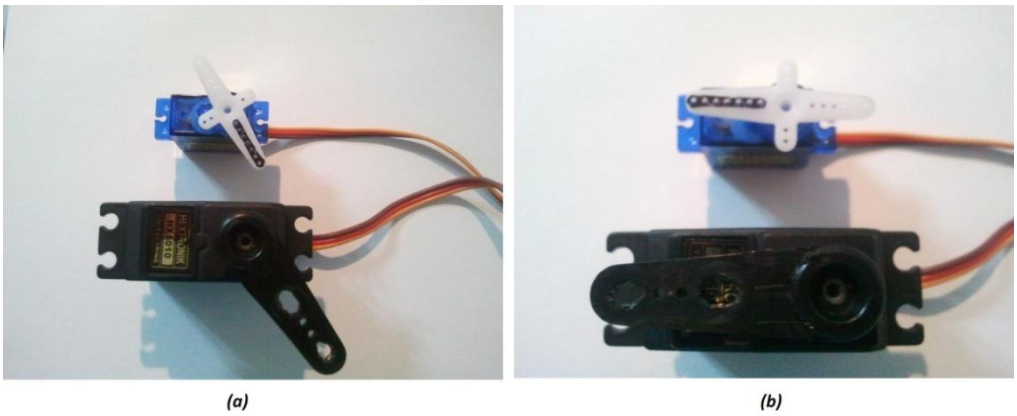


Figura 4.3 - Definição do Ponto Inicial do Eixo do Servomotor.

Fonte - (*O Autor*).

O segundo teste seria descobrir o ponto final do eixo, aproveitando o resultado anterior e com o braço conectado ao eixo já ajustado foi possível verificar o raio de ação dos servomotores, conforme ilustração na Figura 4.4.

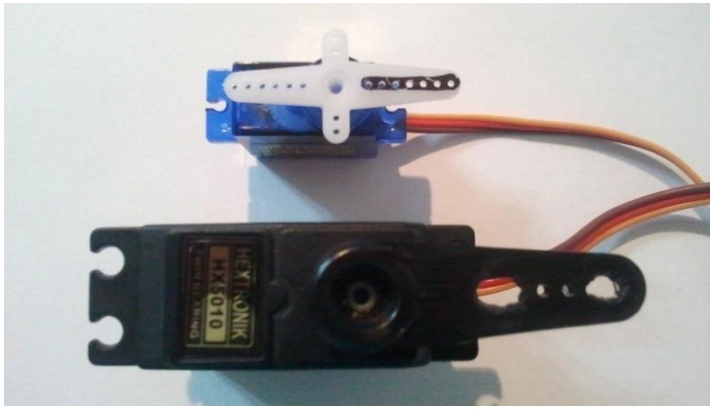


Figura 4.4 - Definição do Ponto Final do Eixo do Servomotor.

Fonte - (*O Autor*).

Após os ajustes iniciais serem realizados e sabendo da capacidade de cada servomotor foi necessário testar exaustivamente tanto as engrenagens plásticas quanto a utilização do

dispositivo e com isso determinar até onde o seu funcionamento não era prejudicado por dano nas partes internas ou por falha de comunicação. O Arduino foi utilizado para controlar os servomotores nos pontos de mínima e máxima e após 55 minutos ficou visível uma variação na rotação do eixo, na qual em diversos momentos ocorria um travamento das engrenagens. Afim de que fosse verificada a origem do erro é que foi necessário desmontar o servomotor e analisar os dentes das engrenagens para descobrir se o teste gerou um dano ao dispositivo, conforme pode ser visto na Figura 4.5.

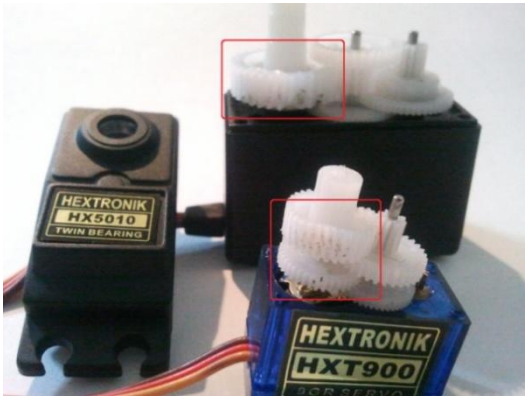


Figura 4.5 - Análise do Interior do Servomotor e Verificação de Danos.

Fonte - (*O Autor*).

Analisando o interior do servomotores foi visível um dano que pode ter sido provocado por um aquecimento ou pela quantidade de movimentos realizados, onde tanto o limitador do eixo quando a parte que limitada o eixo ficaram danificados, com isso é que os servomotores HX12K da mesma fabricante foram eleitos para serem utilizados no decorrer da implementação na qual simulariam os vidros e travas.

Sabendo que as engrenagens dos servomotores HX12K são produzidas em metal e da sua capacidade é que quatro unidades desse modelo foram utilizadas para simular o vidro, a travas e a locomoção.

De posse de informações acerca do ponto de mínima e máxima locomoção e sabendo que o dispositivo pode ser modificado para que realize movimentos maiores que 180º é que as duas unidades utilizadas para simular a locomoção do protótipo foram modificadas. Levando em consideração o que foi anteriormente descrito no item 3.6 acerca de modificação de servomotores e dos resultados obtidos anteriormente é que o potenciômetro e o limitador do eixo foram removidos do sistema afim de que fosse possível realizar movimentos superiores a 360º e com isso a representação de uma roda real.

Após a modificação devidamente realizada e testada foi verificado um erro acerca de seu funcionamento, onde o *feedback* do sistema que era realizado pelo potenciômetro em

conjunto com o limitador não existiam mais e com isso o servomotor opera em regime de loop, com isso ou ele está movimentando para frente ou para trás, mas nunca está parado.

Afim de que o servomotor fosse controlado pela programação e que o erro gerado fosse solucionado é que a função *interrupts()* do Arduino foi utilizada para que o dispositivo ficasse bloqueado após a realização da tarefa. Caso algum valor fosse enviado para realizar uma função é que a função *noInterrupts()* desbloquearia o sistema, liberando-o para a realização das atividades previamente determinadas. Em resumo foi necessário substituir o conjunto potenciômetro e limitador pelas funções de *interrupts()* e *noInterrupts()* para que fosse o controle do servomotor funcionasse de forma correta, como era anteriormente a modificação.

4.3.2. Configuração do *Bluetooth*

O dispositivo BlueSMiRF Gold foi utilizado em conjunto com o Arduino para receber e enviar dados provenientes do Amarino que está instalado no Smartphone com Android. O equipamento referenciado anteriormente possui configurações básicas de fábrica e com isso foi necessário realizar modificações de nome, senha e controle de acesso remoto para que a segurança estivesse em níveis aceitáveis.

A taxa de transmissão padrão do dispositivo é de 115200bps e a mesma não foi modificada nessa etapa de configuração do *Bluetooth*.

Para que fosse possível modificar os itens de segurança definidos anteriormente foi necessário utilizar um adaptador USB para Serial, que permite conectar o *Bluetooth* diretamente ao computador e com isso realizar diversas modificações por meio da Porta Serial e Prompt de Comando, através dos comandos localizados no manual de especificações.

O dispositivo *Bluetooth* foi conectado ao adaptador e sucessivamente ao computador e com isso o mesmo poderia ser acessado utilizando uma porta serial, visto que não seria necessário efetuar nenhuma instalação pelo fato de ser reconhecido automaticamente. Na Figura 4.6 está a representação das etapas anteriormente descritas.

Figura 4.6 - Conexão do *Bluetooth* ao Computador utilizando o adaptador.

Fonte - (*O Autor*).

Após o dispositivo ser reconhecido no computador e sabendo-se qual a porta serial definida para o mesmo é que através do prompt de comando as configurações poderão ocorrer de maneira mais conveniente.

Com o aplicativo iniciado é possível definir as novas configurações para o dispositivo *Bluetooth*. Para acessar as configurações foi utilizado o comando '*$$$*' para acessar as configurações (ver *item (a)* da Figura 4.7), *log*o após o prompt retorna '*CMD*' na qual espera um comando. Para configurar o nome do dispositivo foi utilizado o comando '*SN,JohnnieMonster*' (ver *item (b)*) e após pressionar a tecla *enter* é informado na tela o código '*AOK*' que representa que o comando foi aceito, após modificar o nome foi necessário modificar a senha utilizando o comando '*SP,18201801*' como está descrito no *item (c).*

Para que a segurança fosse definida de forma mais completa foi necessário configurar o endereço do dispositivo remoto para que somente aquele determinado dispositivo tivesse acesso para envio e recebimento de dados com sucesso. Dessa forma, qualquer dispositivo poderia conectar ao *Bluetooth* que está ligado ao Arduino, mas nenhum erro é gerado, permitindo assim somente ao Smartphone cadastrado acesso total a implementação da solução. Para essa modificação foi utilizado o comando '*SR,A07591A52698*' conforme pode ser visto no *item (d).*

Afim de que sejam representadas de forma sucinta as modificações realizadas na configuração do dispositivo é que o comando '*D*' (*item (e)*) foi necessário para informar os diversos parâmetros configurados no referido *Bluetooth*. No *item (f)* é descrito o nome configurado anteriormente, o *item (g)* representa a taxa de transmissão que não foi alterada, o *item (h)* é o modo de operação e o mesmo ficou configurando para trabalhar como '*slave*' porque o Smartphone seria o '*master*' para essa solução. A nova senha configurada pode ser

vista no *item (i)* e o endereço físico do dispositivo remoto está representado no *item (j)* da Figura 4.7.

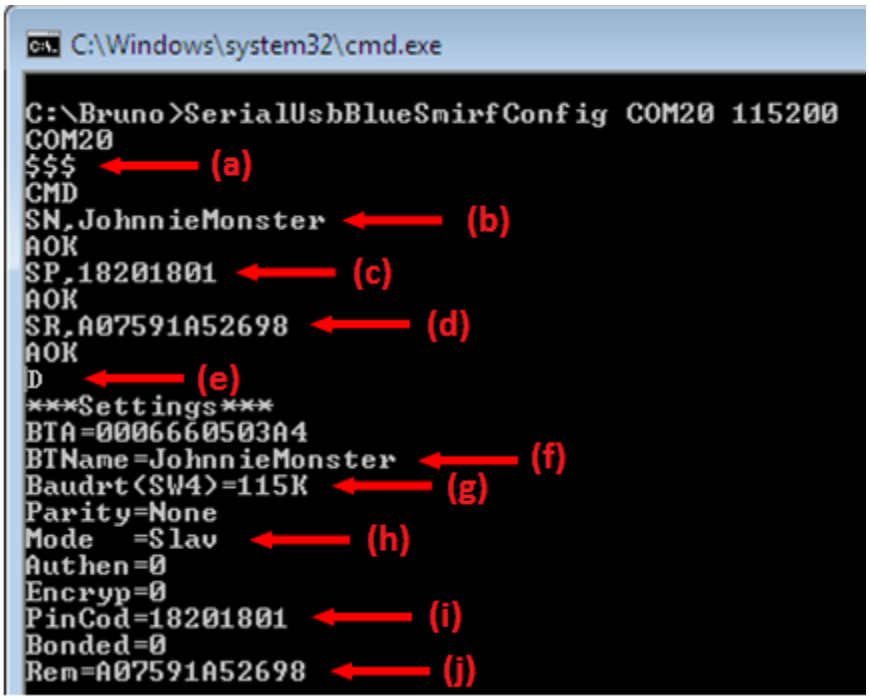


Figura 4.7 - Modificações das Configurações do *Bluetooth*.

Fonte - (*O Autor*).

Todos os comandos utilizados para a realização das modificações foram extraídos do manual de usuários avançados do [ROVING].

4.3.3. Instalação do Amarino

O aplicativo utilizado para a comunicação entre o celular e o Arduino foi obtido através de seu site oficial, [AMARINO]. Na seção de *download* é possível baixar a segunda versão *software* e o *plugin* do Amarino que permite integrar os mecanismos como acelerômetro, bússola digital com os eventos do protótipo.

Após realizar o *download* e acessar a pasta onde os arquivos estão localizados (Figura 4.8) é possível realizar a instalação dos mesmos.

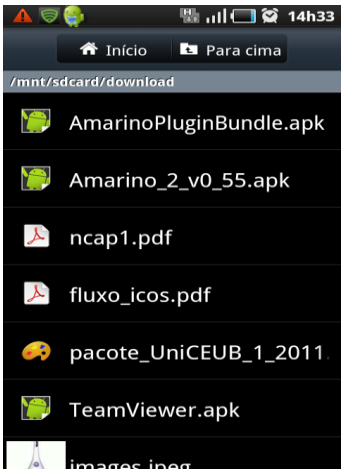


Figura 4.8 - Pasta de Arquivos no Android com os Aplicativos do Amarino.

Fonte - (*O Autor*).

Ao clicar no aplicativo '*Amarino_2_v0_55.apk*' que é referente ao Amarino 2.0.55, aparecerá uma tela informando que o aplicativo será instalado, conforme está descrito no item (*a*) da Figura 4.9. O item (*b*) que é a instalação do *plugin* e foi realizado o mesmo procedimento descrito anteriormente.

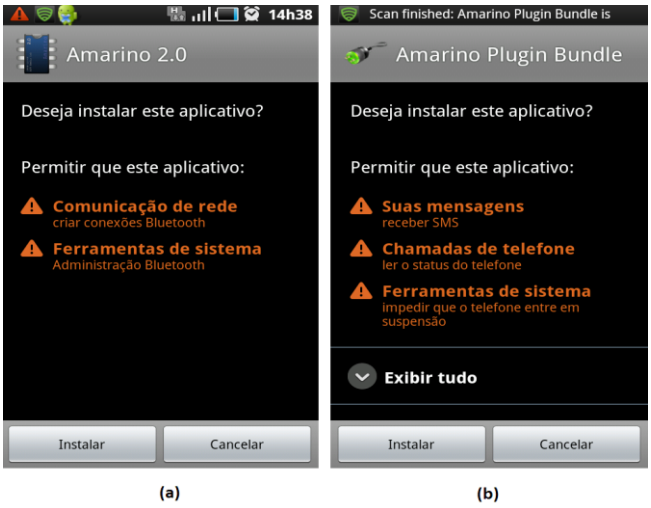


Figura 4.9 - Instalação do Amarino e *Plugin* Bundle no Android.

Fonte - (*O Autor*).

Após a instalação dos aplicativos é possível localizar seu ícone facilmente na tela (*a*), conforme é visto na Figura 4.10. O *plugin* está interligado ao Amarino e somente pode ser acessado através do aplicativo mestre, ou seja, por meio do Amarino 2.0.55. A tela inicial do aplicativo (*b*) contém o dispositivo que está conectado ao Arduino e que anteriormente foi modificado. O *item (c)* mostra o evento cadastrado pra trabalhar em conjunto com o Arduino, no caso o '*Accelerometer Sensor*' que possui a '*ID: A*', esse valor de ID será utilizado mais adiante para cadastrar na programação de onde serão originados os dados dos eventos.

Figura 4.10 - Aplicativo Amarino no Android com Funcionalidades.

Fonte - (*O Autor*).

4.3.4. Programação de Atividades

A linguagem *Wiring* foi utilizada para a programação das atividades. Inicialmente, foi necessário especificar o que seria realizado em cada fase, com isso a programação foi mais bem desenvolvida para que cada item realizasse a atividade com o melhor tipo de aproveitamento e a certificação de que nenhum equipamento realizaria algo além do necessário.

A parte inicial da programação foi testar os servomotores que seriam utilizados no protótipo e com isso definir como seria a atividade que cada um realizaria para simular as diversas atividades alvo desse projeto. Nesse caso o seguinte código foi utilizado para verificar o funcionamento de ângulo a ângulo a rotação do eixo do servomotor conforme está descrito na Figura 4.11.

TesteServo

```
#include <Servo.h>      //Biblioteca do Servomotor

Servo servo1;           //Cria uma variavel para o Servomotor

void setup(){
  servo1.attach(9);     //Define o pino 9 para o Servomotor
}

void loop(){
  for(int pos = 0; pos < 180; pos += 1){ //Faz girar de 0 a 180 graus
    servo1.write(pos);  //Realiza o movimento para frente de 1 a 1
    delay(10);   //Espera 10ms ate a proxima instrucao
  }
  for(pos = 180; pos >= 1; pos -= 1){  //Faz girar de 180 a 0 graus
    servo1.write(pos);  //Realiza o movimento para tras de 1 a 1
    delay(10);   //Espera 10ms ate a proxima instrucao
  }
}
```

Figura 4.11 - Teste de Funcionamento do Servomotor.

Fonte - (*O Autor*).

Após verificar o funcionamento do servomotores de movimento tanto para frente quanto para trás foi necessário definir quais atividades poderiam utilizar esse princípio, com isso a simulação de vidro e trava elétrica seriam mais bem aproveitadas visto que o seu funcionamento é dado pela seguinte forma:

- **Vidro Elétrico:** para o funcionamento dessa atividade no caso de vidro aberto ou fechado foram utilizadas duas abordagens. Para abrir o vidro seria necessário girar a roldana acoplada ao servomotor e com isso a realização do movimento que é o descer a engrenagem e com isso a realização do movimento e para fechar o vidro o movimento seria o inverso. Levando em consideração os testes realizados anteriormente é que a programação está baseada e com isso o movimento de 0 a 180 graus é para abrir os vidros e de 180 a 0 grau é para fechar os vidros, o item 4.3.5.1 descreve o funcionamento da estrutura de funcionamento dessa atividade.
- **Trava Elétrica:** assim como a descrição anterior é que essa funcionalidade foi desenvolvida, em que um braço está conectado em uma das extremidades ao eixo do servomotor e na outra a estrutura que simula a trava, com isso tem-se que para abrir é necessário ter o mesmo movimento realizado pela estrutura do vidro elétrico e o item 4.3.5.2 descreve a construção dessa etapa.

Após a definição dos movimentos de simulação dos vidros e travas foi necessário desenvolver a lógica de funcionamento do Smartphone com o protótipo e com isso definir como seria a comunicação de um sistema para com o outro.

Conforme descrito anteriormente no item 4.3.3 e que ao final do processo o Amarino está instalado no Smartphone e com capacidade de comunicação com o *Bluetooth* do protótipo é que o teste inicial foi realizado, seguindo os passos iniciais do funcionamento do Amarino.

Para fazer o teste inicial foi necessário configurar um evento no Smartphone para que um valor seja enviado para o Arduino no momento em que a conexão esteja ativa. No *item (a)* da Figura 4.12 o evento marcado em vermelho é o que será utilizado para testar a comunicação e com isso definir como é o funcionamento da atividade. O *item (b)* é a descrição do funcionamento da atividade ora selecionada anteriormente e como os dados serão visualizados na tela e o *item (c)* é a definição do identificador para esse evento.

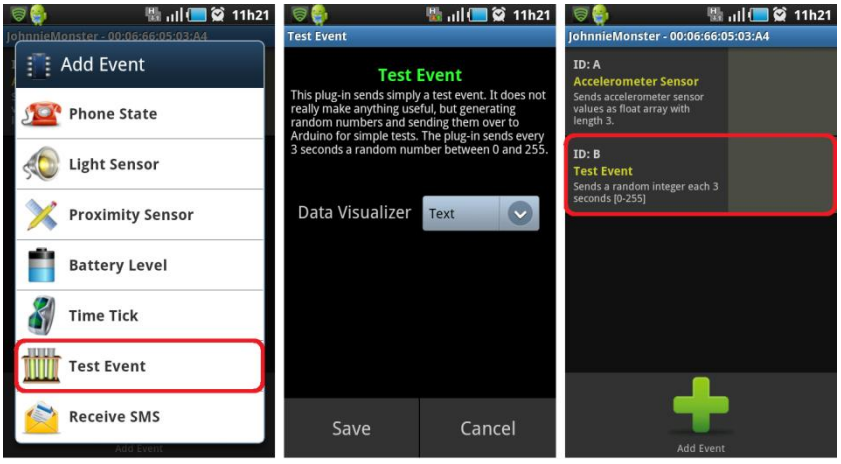


Figura 4.12 - Evento de Teste no Amarino.

Fonte – (*O Autor*).

No Arduino IDE existe uma infinidade de exemplos criados para testes e para iniciação de projetos e o Amarino disponibiliza uma biblioteca nomeada '*MeetAndroid*' que é capaz de comunicar um Smartphone com Android e Arduino. Na Figura 4.13 abaixo está o código de exemplo utilizado para testar a comunicação dos dispositivos e verificar como é o funcionamento da programação que será utilizada para controle do protótipo.

PiscaLED

```
#include <MeetAndroid.h>  //Biblioteca do Amarino

MeetAndroid meetAndroid;  //Variavel do Amarino
int onboardLed = 13;  //Define o pino para testes no LED

void setup(){
  Serial.begin(115200); //Taxa de Transmissao
  meetAndroid.registerFunction(testEvent, 'B'); //ID cadastrada no Smartphone
  pinMode(onboardLed, OUTPUT);  //Indica que o pino recebera valores externos
  digitalWrite(onboardLed, HIGH);  //Acende o pino para iniciar o processo
}

void loop(){
  meetAndroid.receive(); //Recebe os Eventos do Smartphone
}

void testEvent(byte flag, byte numOfValues){
  flushLed(300);  //Chama a funcao que pisca o LED
  flushLed(300);  //Chama a funcao que pisca o LED
}

void flushLed(int time){
  digitalWrite(onboardLed, LOW);  //Apaga
  delay(time);  //Tempo de Espera
  digitalWrite(onboardLed, HIGH); //Acende
  delay(time);  //Tempo de Espera
}
```

Figura 4.13 - Teste de Comunicação entre Smartphone e Arduino.

Fonte - (*O Autor*).

A Figura 4.13 descreve a atividade que será realizada, no caso o LED piscar conforme as dados são recebidos do Amarino. A variável '*meetAndroid*' foi criada para definir um método de funcionamento da biblioteca, onde a mesma define todos os parâmetros de configuração tanto de envio como de recebimento dos dados.

A taxa de transmissão foi de 115200bps conforme configuração realizada anteriormente no item 4.3.2. A função '*registerFunction*' define o nome da função que realizará a atividade e o identificador '*B*' corresponde ao evento cadastrado no item (c) da Figura 4.12. Para um simples teste foi configurado o pino 13 para piscar o LED e seu nome foi definido para '*onboardLed*' e o mesmo espera valores externo e por padrão estará sempre aceso.

A estrutura '*void loop()*' é muito simples e define a rotina que executará o processo inúmeras vezes. A variável '*meetAndroid*' está configurada com a função '*receive()*' e com isso é possível receber os dados provenientes do evento do Amarino.

A função '*testEvent*' é configurada em dois momentos, inicialmente é definida qual a ID cadastrada no Amarino para ser a origem dos dados e posteriormente para a atividade propriamente dita, na qual a mesma necessita de dois parâmetros, o primeiro é a '*flag*' e a

segunda os dados dessa *flag*, nesse caso, se a *flag* '*B*' for a mesma que está definida na programação da atividade o processo funcionará, caso contrário não. Com isso é possível definir inúmeras atividades no celular e para cada uma delas é possível criar um funcionamento diferenciado. Já a função '*flushLed*' representa a funcionalidade propriamente dita e que fará o LED piscar no Arduino.

Após realizar o *upload* do código fonte para o Arduino e conectar o *Bluetooth* do Arduino com o do Smartphone é possível testar o funcionamento da atividade, na qual o LED está apagado quando não recebe os dados *(a)* e após receber as informações o mesmo fica aceso *(b)*, conforme pode ser visualizada na Figura 4.14.

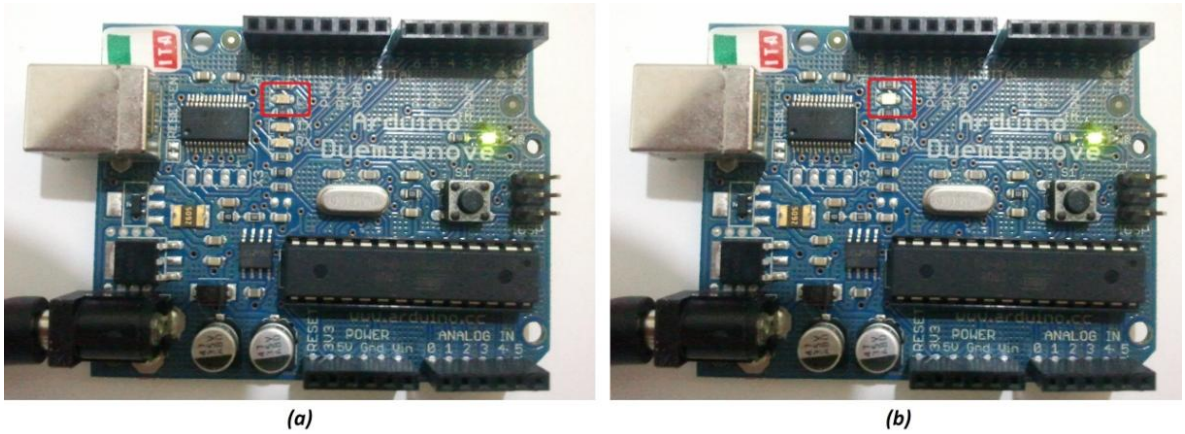


Figura 4.14 - Arduino Piscando os LED com a utilização do Amarino.

Fonte - (*O Autor*).

Analisando o código acima e o funcionamento do servomotor realizado anteriormente foi possível definir como será a atividade alvo desse projeto. Para a programação do protótipo foi utilizados os conceitos referenciados acima e com isso utilizá-los em conjunto para a realização da atividade de vidros e travas elétricas.

4.3.5. Montagem do Protótipo

O protótipo foi desenvolvido em partes e para que cada processo fosse concluído antes de prosseguir com a implementação foram realizados testes diferenciados para comprovar o funcionamento de cada estrutura, ou seja, para a construção da simulação de vidros elétricos foram realizados diversos testes acerca de seu funcionamento e com isso foi possível adicionar correções durante sua montagem assim como na simulação de travas elétricas.

4.3.5.1. Estrutura de Simulação do Vidro Elétrico

Para construir a estrutura que faria a função de simular o vidro elétrico foi inicialmente utilizado a idéia de adaptar algum dispositivo já existente e que possuía

características simulares ao funcionamento do referido sistema em questão. Dos diversos equipamentos testados para ao desenvolvimento da estrutura o que mais se assemelha ao produto final é uma unidade de CD/DVD, independentemente da funcionalidade de leitora ou gravadora da mesma. No caso os equipamentos utilizados do referido dispositivo foram: a bandeja de CD/DVD e a roldana que encaixa a bandeja a estrutura da unidade. Para representar o vidro foi usado uma peça em acrílico do tamanho 5 cm x 7 cm x 4 mm e o servomotor representaria a máquina do vidro para abrir e fechar o vidro, conforme pode ser visto na Figura 4.15.

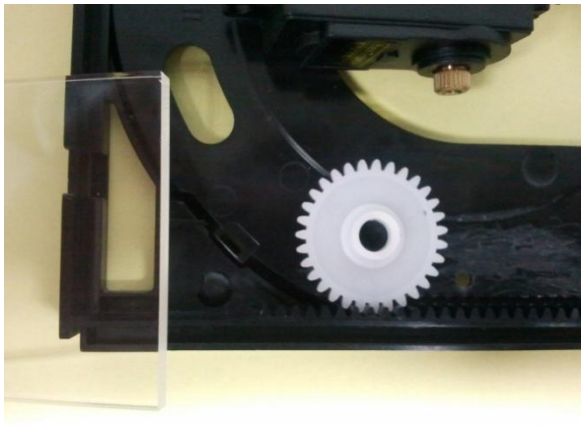


Figura 4.15 - Estrutura inicial da Simulação de Vidro Elétrico.

Fonte - (*O Autor*).

Após realizar o levantamento dos equipamentos a serem utilizados para a montagem foi necessário definir como o servomotor seria acoplado à placa de acrílico que representaria o vidro e a forma de encaixe tanto do simulador do vidro na bandeja como da estrutura final a placa de acrílico que representa o protótipo. A idéia inicial foi afixar o servomotor como pode ser visto na Figura 4.16, mas por questões de funcionamento e falta de confiabilidade da afixação foi definido que o dispositivo seria utilizado da forma que está apresentada na Figura 4.17. Para que não fosse um item muito grande e de difícil afixação ao protótipo foi necessário cortar a peça que compõe a bandeja e utilizar um pedaço da mesma para servir como gabarito da roldana e dessa forma agir como trava para que fosse realizado nenhum movimento fora do espaço delimitado.

Figura 4.16 - Definição do Posicionamento do Servomotor.

Fonte - (*O Autor*).

Após a finalização da estrutura montada para simular o vidro elétrico foi necessário utilizar uma peça em latão para unir a estrutura da bandeja com acrílico ao servomotor. A peça em latão foi dobrada para servir de gabarito da estrutura e também para o conjunto ficar preso ao restante do protótipo, como pode ser visualizado na Figura 4.17.

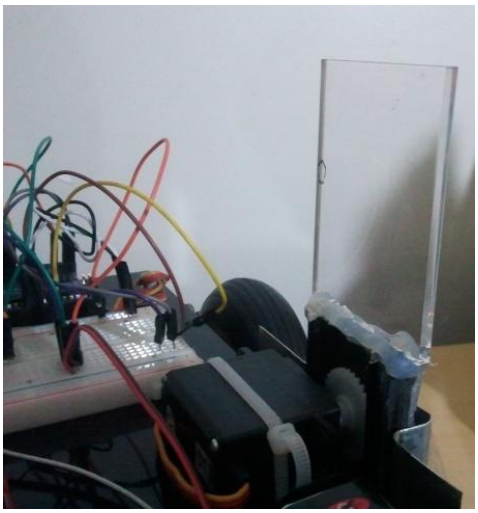


Figura 4.17 - Estrutura montada para realização de testes.

Fonte - (*O Autor*).

Com toda a estrutura pronta e bem afixada foi possível testar o seu funcionamento e delimitar na programação o movimento de abrir e fechar o vidro. Como a estrutura da bandeja não é lisa e possui algumas lacunas por onde poderia ocorrer o travamento da roldana foi utilizada uma pistola elétrica de cola quente de bastão para que essas brechas fossem preenchidas e assim a prevenção de alguma falha no funcionamento.

4.3.5.2. Estrutura de Simulação da Trava Elétrica

Para simular a trava elétrica foi utilizado o conhecimento acerca do funcionamento dos servomotores e dos diversos resultados obtidos com o mesmo.

Levando em conta que o funcionamento seria simples e que a real função de uma trava é somente abrir e fechar um pino com a utilização de uma haste de metal para a realização do movimento é que a definição desta estrutura foi baseada. Na qual consiste em criar uma estrutura para servir como gabarito e que é onde o pino da trava está conectado e caso um movimento seja realizado para baixo ou para cima é que refletirá uma ação no objeto referenciado anteriormente.

Para que a simulação funcione conforme o sistema original foi definido que se o braço conectado ao servomotor estiver no ponto de 0º, representará a função de trava fechada e caso ocorra à movimentação do mecanismo e o valor do ângulo for maior que 90º é que representará a função de trava aberta.

Utilizando os equipamentos que restaram da gravadora de DVD após a construção da estrutura de vidros elétricos e um tubo de uma caneta de plástico foi possível usar essas peças em conjunto e com a ajuda de um arame para conectar o pino que está interligado a estrutura ao braço do servomotor. Na Figura 4.18 está a estrutura definida anteriormente utilizada para testes de funcionamento.

Figura 4.18 - Vista Frontal da Estrutura de Simulação da Trava Elétrica.

Fonte - (*O Autor*).

Na Figura 4.19 está a vista traseira da estrutura montada para a simulação de trava elétrica. Com a simplicidade na construção foi possível obter o resultado esperado sem ser necessário despender muito tempo para o seu desenvolvimento.

Figura 4.19 - Vista Traseira da estrutura montada.

Fonte - (*O Autor*).

5. ANÁLISE DO MODELO PROPOSTO

5.1. Aplicação do Protótipo Proposto.

O protótipo tem por objetivo realizar quase que a mesma função de uma chave automotiva do tipo canivete, ver Figura 5.1. A sua aplicação é em caráter de apoio, levando em consideração ao problema exposto no item 2.1. Com isso, pode-se inferir que sua funcionalidade ainda é para fins acadêmicos, porque nos diversos testes realizados no protótipo ficou claro que alguns itens apresentação falhas no momento da aplicabilidade.

Através dos resultados obtidos e dos custos com todos os equipamentos utilizados para a montagem do protótipo é notável que o mesmo possua um valor exorbitante, levando em consideração os equipamentos e tecnologias utilizadas que serão descritas adiante.

No referido projeto é considerado válido se ao final da implementação o mesmo realizar as funcionalidades descritas no item 1.4, no caso o acionamento dos Vidros (abrir e fechar) e Travas Elétricas (travar e destravar) por meio do Smartphone com Android e da comunicação *Bluetooth*.

Figura 5.1 - Chave do Tipo Canivete – Modelo Audi.

Fonte - (*http://produto.mercadolivre.com.br/MLB-164035548-audi-carcaca-completa-chave-canivete-audi-3-botoes-_JM*).

5.2. Descrição da Aplicação do Protótipo

O Smartphone com Android possui um aplicativo previamente instalado, denominado *Amarino* e por meio deste é possível acionar as funcionalidades do protótipo. Para que seja possível enviar e receber dados do celular e o carrinho é necessário que os dispositivos estejam emparelhados a partir da conexão *Bluetooth*, na qual cada equipamento conhece a outra parte, como endereço físico, senha e que a taxa de transmissão seja a mesma, permitindo assim o tráfego de dados sem nenhum problema.

Após ocorrer o emparelhamento dos dispositivos é possível movimentar o dispositivo celular para frente para baixo (*a*), frente para cima (*b*), esquerda para cima (*c*) ou direita para cima (*d*) e com os valores oriundos desses movimentos é que uma atividade poderá ser executada, graças ao acelerômetro que utiliza os eixos X, Y e Z para informar com precisão o posicionamento do Smartphone, ver Figura 5.2 com os possíveis movimentos do dispositivo descrito anteriormente. De posse dessas informações e do identificador do evento cadastrado no aplicativo *Amarino* referente ao acelerômetro é que ocorre o envio dos dados do celular para o módulo *Bluetooth*, que repassa ao microcontrolador e após análise indica se alguma atividade será executada, seja abrir e fechar os vidros ou travar e destravar os vidros ou não executar nenhuma delas, e após esse processo é enviado para o Smartphone uma mensagem contendo a descrição do evento.

Figura 5.2 - Posicionamento do Smartphone Samsung Galaxy S.

Fonte - (*O Autor*).

5.3. Resultados do Projeto

5.3.1. Resultados Esperados

O resultado esperado do conjunto *software e hardware* é funcionamento das atividades de vidros e travas elétricas que são simuladas por meio da utilização de servomotores. Com o sistema energizado e a comunicação *Bluetooth* ativa entre os dispositivos é possível realizar o movimento que corresponde à funcionalidade desejada. Após a realização da atividade é enviada uma mensagem que reafirma o funcionamento da mesma, comprovando que o movimento previamente realizado é o mesmo cadastrado na programação do dispositivo.

Uso dinâmico do celular para realização de atividades e com isso permitir ao usuário maior facilidade na sua utilização.

Na utilização do *software Amarino* é esperado que os dados enviados e recebidos possam ocorrer sem falhas e que o comando correto seja transmitido para o microcontrolador para posterior análise.

Nos itens de *hardware* são esperados que o módulo *Bluetooth*, Arduino e servomotores possam realizar o funcionamento correto da atividade, que após os dados recebidos do aplicativo e devidamente tratatos por meio do código compilado no microcontrolador o mesmo corresponda à atividade programada e após esse processo informe ao *Amarino* que tudo foi realizado com sucesso.

5.3.2. Resultados Obtidos

Com o protótipo implementado e testado foi possível verificar o seu funcionamento através da realização das atividades previamente cadastradas. Com os dados oriundos do acelerômetro e enviados pelo aplicativo Amarino para o conjunto módulo *Bluetooth* e analisado pelo *Arduino* com o auxílio do microcontrolador *ATmega328P* foi possível mensurar a validade do funcionamento que apresenta níveis satisfatórios de aplicabilidade. Ainda assim foi necessário modificar a programação do dispositivo Arduino porque a mesma estava apresentando falhas e em alguns casos não era possível executar algumas funções. Já o *software Amarino* foi crucial para o projeto, por fornecer a interface de comunicação do celular com o microcontrolador, mas diversos testes comprovaram uma insatisfação em determinados momentos, como falhas na aplicação *(a)*, erro de conexão dos dispositivos *(b)* conforme é visto na Figura 5.3.

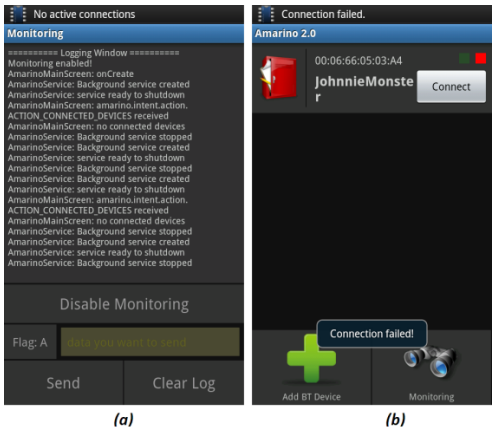


Figura 5.3 - Erros do Aplicativo Amarino no Smartphone Samsung Galaxy S.

Fonte - (*O Autor*).

Com a constante utilização do Smartphone é possível determinar o movimento preciso da funcionalidade determinada, fator que "limita" o projeto no quesito de ser dinâmico, porque em diversos testes realizados ficou comprovada uma espécie de falha que foi

contornada com a adição de algumas comparações na programação das atividades. Para que o problema fosse solucionado de fato seria necessário desenvolver um *software* em Android na qual o *touchscreen* seria utilizado para seleção das atividades ao invés do acelerômetro.

Durante a implementação foram utilizados dez servomotores de variadas configurações, afim de que exaustivos testes comprovassem o funcionamento das atividades e com isso foi possível realizar algumas modificações acerca da velocidade de rotação, angulação do giro do eixo e calibração do movimento nas engrenagens das simulações de vidros e travas elétricas. Para a versão final do protótipo foram selecionados quatro servomotores da marca Hextronik, modelo HX12K, por possuírem engrenagens de metal e com isso uma garantia de que nenhum movimento poderia danificar os equipamentos.

Nos dois servomotores modificados utilizados para a simulação da locomoção do protótipo foi detectado um erro que teve de ser contornado via programação, assim que a bateria era conectada ao microcontrolador, os servomotores começavam a girar sem que nenhuma atividade estivesse ativa. No momento em que o celular estava emparelhado ao protótipo, os servomotores realizavam suas funcionalidades de forma correta, mas ao encerrar a comunicação entre os dispositivos e caso alguma informação estivesse sendo transmitida é que ocorria a falha, na qual os mesmos continuavam a realizar atividades sem que fosse necessário. Nesse caso foi necessário utilizar a função *interrupts()* do Arduino, que informa ao "sistema" que todas as funções estão interrompidas até que a função *noInterrupts()* libere para a realização de atividades, todo esse problema foi ocasionado porque o potenciômetro foi substituído e o limitador da engrenagem foi removido, na qual eram utilizados para informação ao circuito de controle do servo os pontos iniciais e finais da engrenagens.

Os sistemas que simulam os vidros e travas elétricas funcionam de forma precisa, levando em consideração os equipamentos utilizados para o desenvolvimento dos mesmos, no caso uma bandeja da gravadora de DVD e as engrenagens que fazem parte do seu funcionamento.

5.3.3. Comparação entre Resultados Esperados e Obtidos

A combinação de *software* e *hardware* e as tecnologias envolvidas provam que os resultados foram satisfatórios e produziram a solução propriamente dita, porque com a utilização do *Amarino* no Smartphone com Android, módulo *Bluetooth* e Arduino foi possível realizar o controle das funções de vidros e travas elétricas e ir mais além, com testes na locomoção do protótipo.

Considerando a precisão do acelerômetro do celular e a constante utilização do dispositivo com o protótipo, proporciona a facilidade e dinamicidade de uso, porque após incansáveis testes de funcionamento é definir que o movimento necessário para a realização da atividade.

Com relação ao *software* Amarino é esperado que algumas falhas possam ocorrer, justamente porque o mesmo não foi idealizado para esse projeto e sim em uma escala mais ampla de utilização, mas como o fator tempo e em diversos momentos durante a implementação ocorreu que equipamentos foram danificados, problemas com importação de produtos e a programação teve de ser modificada em diversos momentos é que sua utilização foi imprescindível para a conclusão do projeto, justificando que os "problemas" encontrados deveriam ser solucionados por meio de programação de diversas rotinas a fim de prevenção de possíveis falhas no funcionamento.

5.4. Custos do projeto

Inicialmente o projeto tinha o objetivo de ser de baixo custo, por utilizar *software* livre e não ser necessário utilizar uma licença de funcionamento e também por representar algumas funcionalidades automotivas de '*pouco valor*' considerando o preço de um veículo, mas isso não foi possível porque os valores exorbitantes dos dispositivos no mercado brasileiro provam o contrário.

O custo de um projeto é um fator muito importante seja em um ambiente acadêmico ou corporativo e o planejamento no momento da aquisição dos equipamentos foi necessário e levado em consideração em diversos momentos. Foram utilizados servomotores, conectores, microcontroladores, baterias, placas de acrílico, gravadoras de DVD e rodas.

O orçamento total do projeto está representado na Tabela 1 com descrição dos equipamentos utilizados e custos.

Tabela 1 – Custo Total com os Equipamentos utilizados no decorrer do Projeto para a construção de uma unidade do protótipo.

Quantidade	Descrição	Origem	Valor Unitário	Valor Total
2	Arduino Duemilanove	Robocore	R$ 99,00	R$ 198,00
1	Arduino Kit Iniciante V3.0	Robocore	R$ 239,00	R$ 239,00
1	Arduino *Bluetooth*	Robocore	R$ 399,00	R$ 399,00
2	Servomotor Hextronik HXT900	Mercado Livre	R$ 10,00	R$ 20,00
3	Servomotor Hextronik HXT5010	Mercado Livre	R$ 26,00	R$ 78,00
4	Servomotor Hextronik HX12K	Mercado Livre	R$ 40,00	R$ 160,00
1	Servomotor Tower Pro Mg995	Mercado Livre	R$ 35,00	R$ 35,00
1	Módulo BlueSMiRF Gold	Lab de Garagem	R$ 203,00	R$ 203,00

5	Baterias Duracell 9V	Dantenas	R$ 14,00	R$ 70,00
1	Placa de Acrílico	Carplac	R$ 70,00	R$ 70,00
2	Rodas de Aeromodelismo DUBR 350 TL (Par)	Salles Hobby	R$ 45,00	R$ 90,00
2	Protoboard	Contato Eletrônica	R$ 15,00	R$ 30,00
2	Conjunto de Jumper Premium	Robocore	R$ 19,00	R$ 38,00
1	Itens Eletrônicos (LED, Diodo, Resistor, Cabos)	Contato Eletrônica	R$ 160,00	R$ 160,00
1	*Software* Amarino 2.0.55	Amarino Toolkit	R$ 0,00	R$ 0,00
1	Placa FTDI 5V - USB para Serial	*SparkFun*	R$ 53,00	R$ 53,00
1	Fonte de Alimentação Regulável - 5V/3.3V	*SparkFun*	R$ 35,00	R$ 35,00
2	Gravadora de DVD com Defeito	Biotech 203 Norte	R$ 5,00	R$ 10,00
1	Smartphone Samsung Galaxy S	Vivo	R$ 850,00	R$ 850,00
			Total Gasto no Projeto Final:	R$ 2.738,00

Fonte: (*O Autor*)

O orçamento poderia ter um valor menor, porque com a quantidade de equipamentos reservas era possível montar um novo protótipo. O dispositivo Arduino *Bluetooth* na Loja Multilógica custa R$ 614,00, que é um valor exorbitante se for comparado ao valor gasto conforme está descrito acima.

O custo total poderia diminuir bastante se todos os produtos fossem adquiridos na *SparkFun Eletronics*, mas o maior problema é acerca da importação, pois itens como a Placa FTDT 5V – USB para Serial e Fonte de Alimentação Regulável 5V/3.3V foram adquiridos no dia 12/05/2011 e demoraram cerca de vinte dias para serem entregues, pelo fato de permanecerem retidos na Alfândega Brasileira durante doze dias sendo que o frete contratado foi o de entrega internacional expressa que gasta no máximo dez dias úteis. Isso comprova a ineficiência do Brasil, onde equipamentos e tecnologias modernas demoram e custam muito para o consumidor brasileiro, seja ela por ineficiência dos órgãos de fiscalização ou pela quantidade de impostos pagos para importação de produtos. O Módulo BlueSMiRF Gold foi adquirido através da Loja Virtual Lab de Garagem, que é uma parceira oficial da SparkFun Eletronics e que todos os meses realiza importações, na qual diminui os custos e beneficia o consumidor, pelo fato do projeto dessa loja apoiar a idéia de projetos de garagens. Esse mesmo produto se fosse adquirido na Multilógica custaria cerca de R$ 306,00, nesse caso seria possível comprar quase dois equipamentos idênticos, reforçando a idéia da quantidade de impostos embutidas nos equipamentos.

A aquisição dos diversos equipamentos é um fator imprescindível para o projeto, independente da plataforma utilizada é necessário planejar custos, realizar diversos orçamentos para que os gastos não sejam abusivos ou desnecessários. Os dispositivos adquiridos para o projeto possuem um valor elevado se for comparado a outras plataformas, justamente porque o Brasil é um país muito carente de tecnologia de ponta e as empresas

existentes cobram uma fortuna pelos produtos importados. Importar produtos é bom, o preço pago é quase metade do valor cobrado no Brasil, mas o fator tempo de entrega, alfândega e impostos são variáveis quase incontroláveis, porque dependendo do valor o mesmo pode ser taxado ou não e com isso seria necessário gastar mais um montante para poder receber o produto. Outro fator que necessariamente deve ser mencionado é acerca dos órgãos públicos, como Alfândega e Correios, que demoram e muito para verificação e liberação dos produtos e também na entrega, isso prova a ineficiência do sistema como um todo, onde o consumidor é o que mais sofre com atrasos, impostos e valores abusivos.

6. CONCLUSÃO

6.1. Conclusões

A proposta para o projeto surgiu a partir da vivência prática em relação ao item 1.1, a frustração de ter o veículo fechado com a chave em seu interior, aliado ao fato de que, atualmente, o celular é um dispositivo que faz parte da vida de muitas pessoas e possui diversas funcionalidades. O Android foi utilizado, neste projeto, porque é uma tecnologia atual, que permite a criação de uma infinidade de utilizações. A plataforma Arduino está em constante evolução e por ser open source é possível modificá-la para criar o que se deseja com o auxílio de simples ferramentas. A união dos itens mencionados proporcionou a construção do protótipo na qual é possível controlar os vidros e travas elétricas por meio de um Smartphone com Android e aplicativo *Amarino* utilizando a comunicação *Bluetooth* para envio dos dados do celular para o Arduino e vice-versa.

Com os equipamentos adquiridos foi possível realizar os testes iniciais para testar os funcionamentos dos diversos dispositivos e definir qual função cada um realizaria. Testes de segurança e de correção de erros foram utilizados para melhorar cada funcionalidade e com isso mensurar se cada atividade estava funcionamento de forma correta para validação dos resultados.

Os resultados obtidos comprovam o funcionamento das atividades que simulam os vidros e travas, na qual equipamentos simples foram utilizados para a realização destas funcionalidades.

Alguns problemas foram enfrentados, e com a utilização da programação foi possível corrigir tais erros, para que a solução final atendesse aos requisitos definidos. Um item que superou as expectativas foi o aplicativo Amarino, que por não ter sido desenvolvido para a solução aqui apresentada, foi capaz de proporcionar o resultado final do projeto, que era utilizar o celular e o *Bluetooth* para controle de itens automotivos conforme descrito no item 1.4. da monografia.

A segurança do aplicativo não existe, qualquer pessoa com acesso ao Smartphone pode utilizar o Amarino e conectar no dispositivo *Bluetooth* e realizar o controle das atividades, para que esse problema fosse diminuído foi necessário cadastrar uma senha para desbloquear a tela do celular, configurar o *Bluetooth* do protótipo com senha de oito dígitos e bloqueio por endereço MAC, na qual somente o dispositivo celular utilizado no projeto tem acesso ao *hardware*. E para a questão de erros foi necessário modificar a programação em diversos momentos, tornando-a mais ágil, mas com um aumento no número de rotinas a

serem executadas. Todos esses problemas poderiam ser solucionados com o desenvolvimento de um aplicativo em Android, específico para o controle das funcionalidades ora descritas anteriormente.

Para a construção do projeto foi necessário realizar diversas pesquisas em meios eletrônicos, livros e manuais de especificação técnica, para que uma gama extensa de opções pudesse ser utilizada para a solução de cada dificuldade enfrentada no decorrer da implementação. A programação das atividades a serem realizadas pelo protótipo foi fundamental para o bom andamento, na qual é onde está à definição das atividades a serem realizadas pelos equipamentos e caso algum recurso fosse utilizado de forma incorreta poderia gerar uma solução que não seja o que foi definido. A realização de testes é fundamental, nessa etapa é possível mensurar se o que foi programado está executando de forma correta, testes de segurança e de verificação de erros foram utilizados para diminuir uma parte dos problemas e melhorar o funcionamento do protótipo como um todo. A aquisição de equipamentos também foi um problema tanto de tempo necessários para serem entregues como de valor total gasto.

A conclusão acerca desse projeto é que um valor muito alto foi gasto para a solução de um problema relativamente baixo, ou seja, os equipamentos reais de vidros e travas elétricas possuem valores irrisórios se comparados aos utilizados na solução em geral, mas levando em consideração que o produto final de *software* e *hardware* combinados cumpriu o escopo do projeto é possível inferir que o mesmo pode ser utilizado para a solução real, desde que um aplicativo específico seja desenvolvido conglomerando senha de acesso para controle de atividades. E se o mesmo for produzido em larga escala à relação custo/benefício será um fator favorável, na qual o proprietário do veículo pode destravar o mesmo sem o auxílio da chave de ignição automotiva, mas com a utilização do Smartphone.

6.2. Sugestão de Futuros Projetos

Conforme descrito anteriormente e levando em consideração uma pequena bagagem acerca dos diversos problemas enfrentados no decorrer da implementação é possível descrever que um aplicativo específico poderia ser desenvolvido e assim controlar as diversas funcionalidades existentes em um automóvel. Com isso a tela *touchscreen* poderia ser mais bem aproveitada para controle e seleção de atividades, como abrir os vidros de acordo com a necessidade do usuário do aplicativo ou outras funcionalidades, como alarme, teto solar, som, iluminação e suspensão a ar caso o automóvel utilizado para o projeto possua.

Com esses novos itens descritos anteriormente combinados com a utilização de um aplicativo no Smartphone é que poderia aproveitar ao máximo as diversas funções que esses produtos proporcionam para o consumidor final, ou seja, a busca por conforto.

REFERÊNCIAS

Livros:

BANZI, Massimo. Getting Started with Arduino. 3ª edição. Estados Unidos: Editora O'Reilly Media/Make, 2008. 128 p.

FOROUZAN, Behrouz A. Comunicação de Dados e Redes de Computadores. 3ª edição. São Paulo: Editora Bookman. 2006. 840 p.

FRANCISCO, Antônio. Motores Elétricos. 2ª edição. Portugal: Editora ETEP - Edições Técnicas e Profissionais, 2008. 180 p.

GIMENEZ, Salvador Pinillos. Microcontroladores 8051. 1ª edição. São Paulo: Editora Person Education do Brasil, 2002. 253 p.

GUIMARÃES, ALEXANDRE de Almeida. Eletrônica Embarcada Automotiva. 1ª Edição. São Paulo: Editora Érica, 2007. 326 p.

LECHETA, RICARDO Rodrigues. Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK. 2ª Edição. São Paulo: Editora Novatec, 2010. 608p.

NICOLOSI, Denys Emílio C. Microcontrolador 8051 Detalhado. 5ª edição. São Paulo: Editora Érica, 2004. 227 p.

TANEMBAUM, Andrew S. Redes de Computadores. 4ª edição. São Paulo: Editora Campus, 2003. 955 p.

Internet:

Arduino – ArduinoBoard*Bluetooth*.

Disponível em: <http://www.arduino.cc/en/Main/ArduinoBoard*Bluetooth*>.

Acesso em: 07 de jun. 2011.

Arduino – Arduino UNO.

Disponível em: <http://arduino.cc/en/Main/ArduinoBoardUno>.

Acesso em: 07 de jun. 2011.

Arduino – Arduino Duemilanove.

Disponível em: <http://www.arduino.cc/en/Main/ArduinoBoardDuemilanove>.

Acesso em: 07 de jun. 2011.

Arduino *Bluetooth* - *SparkFun Electronics*.

Disponível em: <http://www.sparkfun.com/products/8255>.

Acesso em: 07 de jun. 2011.

Bluetooth Modem – BlueSMIRF Gold - *SparkFun Electronics*.
Disponível em: <http://www.sparkfun.com/products/582>.
Acesso em: 07 de jun. 2011.

Android Developers.
Disponível em: <http://developer.android.com/>.
Acesso em: 07 de jun. 2011.

Amarino – "Android meets Arduino".
Disponível em: <http://www.amarino-toolkit.net/>.
Acesso em: 07 de jun. 2011.

ROVING. Roving Networks Bluetooth™ Product User Manual. Los Gatos: 2009. 32 p.
Disponível em: <http://www.sparkfun.com/datasheets/Wireless/Bluetooth/rn-bluetooth-um.pdf>.
Acesso em: 07 de jun. 2011.

ATMEL. *ATmega328P* Preliminary Summary. San Jose: 2010. 1,2 p.
Disponível em: <http://www.atmel.com/dyn/resources/prod_documents/doc8161.pdf>.
Acesso em: 07 de jun. 2011.

APÊNDICE A – Código Fonte do Protótipo

```
//INCLUSAO DE BIBLIOTECAS A SEREM UTILIZADAS
#include <MeetAndroid.h>  //BIBLIOTECA QUE CONVERSA COM O AMARINO E ARDUINO

//DEFINE OS PINOS A SEREM UTILIZADOS PELOS MOTORES E BUZZER
#define buzina          13 //DEFINE O BUZZER PARA INFORMACOES SONORAS NO PINO 2     --NAO TESTEI A FUNCIONALIDADE
#define servoTrava    3 //DEFINE O SERVO DO TRAVA COM O PINO 3, QUE E PWM     --PORTA 6 NAO FUNCIONA FULL    -- ERA PORTA 5
#define servoLeft         9   //DEFINE O MOTOR ESQUERDO COM O PINO 9, QUE E PWM       --FUNCIONANDO OK
#define servoRight        10  //DEFINE O MOTOR DIREITO COM O PINO 10, QUE E PWM       --FUNCIONANDO OK
#define servoVidro    11  //DEFINE O SERVO DO VIDRO COM O PINO 3, QUE E PWM     --PORTA 6 NAO FUNCIONA FULL
//#define servoCentral  11 //DEFINE O SERVO CENTRAL COM O PINO 11, QUE E PWM     --NAO TESTEI A FUNCIONALIDADE
#define farol         12  //DEFINE O FAROL ESQUERDO COM O PINO 12, QUE E PWM     --PORTA 6 NAO FUNCIONA FULL
//#define farolRight    13  //DEFINE O FAROL DIREITO COM O PINO 13, QUE E PWM     --PORTA 6 NAO FUNCIONA FULL

char valorStr;

//DEFINE OS VALORES DE LOCOMOCAO
#define ctDesativa  1     //O VALOR 0 PARA DESATIVAR
#define ctFrente    50    //O VALOR 50 É PARA 'ANDAR PARA TRAS'
#define ctTras      254   //O VALOR 254 É PARA 'ANDAR PARA FRENTE'
#define ctAtiva     253   //O VALOR 255 PARA ATIVA

//VARIAVEIS DO PROJETO
MeetAndroid BrunuhSGS;          //DEFINE A FUNCAO QUE TRABALHA COM O AMARINO/ARDUINO - MeetAndroid
int funcID;                     //CRIA UM IDENTIFICADOR PARA A FUNCIONALIDADE A SER ATIVADA, 1 - 'VIDRO OU TRAVA' E 2 - 'LOCOMOCAO'
char funcSelecionada;           //CRIA UMA VARIAVEL PARA SER UTILIZADA NA SELECAO DE FUNCIONALIDADE

//VARIAVEIS COM POSICOES DOS ARRAYS
const int vtAbscissa = 0;  //DEFINE QUE O PRIMEIRO VETOR E O X
```

```
const int vtOrdenada = 1;  //DEFINE QUE O SEGUNDO VETOR E O Y
const int vtCota     = 2;  //DEFINE QUE O TERCEIRO VETOR E O Z

//VARIAVEL COM O ARRAY DOS DADOS DO ACELEROMETRO
float arrayLocomocao[3];        //DEFINE O ARRAY QUE POSSUI OS VALORES DO ACELEROMETRO PARA A FUNCAO DE LOCOMOCAO

//FUNCAO GERAL DE DEFINICAO DOS PARAMETROS DE CONFIGURACAO E TRANSMISSAO DE DADOS
void setup(){
 prototipoStop();                                  //ATIVA A FUNCAO QUE REINICIALIZA TODAS AS FUNCIONALIDADES DO PROTOTIPO
 Serial.begin(115200);                             //CONFIGURA A TAXA DE TRANSMISSAO DO MODEM BLUETOOTH
 BrunuhSGS.registerFunction(amarinoAclrx, 'A');        //DEFINE A FUNCAO DO ACELEROMETRO E DEPENDE DA ID CADASTRADA NO APLICATIVO AMARINO
}

//FUNCAO QUE FAZ O PROTOTIPO FUNCIONAR
void loop(){
 BrunuhSGS.receive();    //RECEBE OS EVENTOS DO AMARINO, COM OS DADOS DO SENSOR DE COMPASSO E ACELEROMETRO
 prototipoStart();
 delay(50);
}

//FUNCAO CRIADA PARA TRABALHAR COM O ACELEROMETRO NO AMARINO, FLAG = ID E OS VALORES DESSE FLAG
void amarinoAclrx(byte flag, byte numOfValues){
 int ctEixo = 0, ct = 0, idxArray = 0;
 int tmhStr = BrunuhSGS.stringLength();      //DESCOBRE O TAMANHO DA STRING RECEBIDA DO AMARINO, VALOR INTEIRO
 char vlrStr[idxArray];                  //DEFINE O TAMANHO DA VARIAVEL PARA RECEBER OS DADOS
 BrunuhSGS.getString(vlrStr);              //CHAMA A FUNCAO QUE RECEBE OS VALORES E SALVA NA VARIAVEL CRIADA ANTERIORMENTE
 String strValores = idxArray;             //SALVA O CONTEUDO NUM TIPO DE DADO STRING, PORQUE PERMITE MANIPULACOES MAIS COMPLEXAS
 for(idxArray = 0; idxArray < tmhStr; idxArray++){        //PERCORRE O ARRAY EM BUSCA DO CARACTERE ';', QUE E O SEPARADOR DOS VALORES DOS EIXOS
   if(vlrStr[idxArray] == ';'){     //SE O CARACTERE ';' FOR ENCONTRADO, SALVA AS VALORES NOS EIXOS CORRESPONDENTES, VAI FAZER O TRATAMENTO PARA X E Y
```

```
    arrayLocomocao[ct] = stringToFloat(strValores.substring(ctEixo, idxArray));        //LE A STRING DO MENOR VALOR ATE O INDICE ONDE FOI ENCONTRADO O ';', DEPOIS CONVERT PARA NUMERO REAIS E SALVA A POSICAO DE MEMORIA CORRETA
    ctEixo = idxArray + 1;
    ct++;
    if(strValores.lastIndexOf(';') == idxArray){      //SE O ULTIMO INDICE DO ARRAY FOR O ';', FAZ O MESMO PROCEDIMENTO ACIMA, VAI FAZER O TRATAMENTO PARA Z
     arrayLocomocao[ct] = stringToFloat(strValores.substring(ctEixo, tmhStr));        //LE A STRING DO PROXIMO VALOR DE Y ATE O MAXIMO VALOR DA STRING, QUE E ';', DEPOIS CONVERT PARA NUMERO REAIS E SALVA A POSICAO DE MEMORIA CORRETA
    }
   }
  }
}

//FUNCAO CONVERTE OS VALORES DE UMA STRING PARA NUMERO REAIS, FUNCAO ENCONTRADA NA INTERNET, COM MODIFICACOES
float stringToFloat(String caracteres){
 char nCaracteres[caracteres.length()];        //CRIA UMA VARIAVEL COM O TAMANHO A STRING RECEBIDA
 caracteres.toCharArray(nCaracteres, sizeof(nCaracteres));
 return atof(nCaracteres);    //RETORNA EM NUMEROS REAIS O VALOR DA STRING
}

//FUNCAO QUE DESABILITA TODAS AS FUNCIONALIDADES DO PROTOTIPO
void prototipoStop(){
 prototipoFreioVT();
 pinMode(buzina, OUTPUT);
 pinMode(farol, OUTPUT);
 delay(50);
}

//FUNCAO PARA COMPARACAO DE VALORES E DETERMINAR QUAL ACAO A SER TOMADA
void prototipoStart(){
 //ABRIR O VIDRO: COMPARACAO DE VALORES DOS EIXOS X E Z
 if(arrayLocomocao[vtOrdenada] <= -2 && arrayLocomocao[vtCota] >= 4){     //PARA ABRIR O VIDRO E NECESSARIO TER VALORES EM X E Z
  BrunuhSGS.send("Abrir o Vidro");     //ENVIA UMA MENSAGEM PARA O AMARINO, INFORMANDO QUAL O PROCEDIMENTO A SER REALIZADO
  noInterrupts();
  vidroAbrir();        //CHAMA A FUNCAO QUE ABRIR O VIDRO
```

```
  //buzinaAtiva();        //CHAMA A BUZINA COM O BARULHO CORRETO
 }
 //FECHAR O VIDRO: COMPARACAO DE VALORES DOS EIXOS -X E Z
 else if(arrayLocomocao[vtOrdenada] >= 2 && arrayLocomocao[vtCota] >= 4){       //PARA FECHAR O VIDRO E NECESSARIO TER VALORES EM -X E Z
  BrunuhSGS.send("Fechar o Vidro");              //ENVIA UMA MENSAGEM PARA O AMARINO, INFORMANDO QUAL O PROCEDIMENTO A SER REALIZADO
  noInterrupts();
  vidroFechar();        //CHAMA A FUNCAO QUE FECHAR O VIDRO
  //buzinaAtiva();        //CHAMA A BUZINA COM O BARULHO CORRETO
 }
 //ABRIR A TRAVA: COMPARACAO DE VALORES DOS EIXOS Y E Z
 else if(arrayLocomocao[vtAbscissa] >= 2 && arrayLocomocao[vtCota] >= 4){    //PARA ABRIR A TRAVA E NECESSARIO TER VALORES EM
  BrunuhSGS.send("Fechar a Trava");     //ENVIA UMA MENSAGEM PARA O AMARINO, INFORMANDO QUAL O PROCEDIMENTO A SER REALIZADO
  noInterrupts();
  travaAbrir();        //CHAMA A FUNCAO QUE ABRIR O VIDRO
  //buzinaAtiva();        //CHAMA A BUZINA COM O BARULHO CORRETO
 }
 //FECHAR A TRAVA: COMPARACAO DE VALORES DOS EIXOS -Y E Z
 else if(arrayLocomocao[vtAbscissa] <= -2 && arrayLocomocao[vtCota] >= 4){       //PARA FECHAR A TRAVA E NECESSARIO TER VALORES EM
  BrunuhSGS.send("Abrir a Trava");      //ENVIA UMA MENSAGEM PARA O AMARINO, INFORMANDO QUAL O PROCEDIMENTO A SER REALIZADO
  noInterrupts();
  travaFechar();        //CHAMA A FUNCAO QUE FECHAR A TRAVA
  //buzinaAtiva();        //CHAMA A BUZINA COM O BARULHO CORRETO
 }/*
 //ANDAR PARA FRENTE: COMPARACAO DE VALORES DOS EIXOS -Y E Z
 else if(arrayLocomocao[vtOrdenada] >= 7 && arrayLocomocao[vtCota] >= 0){     //PARA ANDAR PARA FRENTE E NECESSARIO TER VALORES EM Y E Z
  BrunuhSGS.send("Andar Para Frente");              //ENVIA UMA MENSAGEM PARA O AMARINO, INFORMANDO QUAL O PROCEDIMENTO A SER REALIZADO
  noInterrupts();
  locomocaoTras();        //CHAMA A FUNCAO QUE ANDA PARA TRAS
 }
 //ANDAR PARA TRAS: COMPARACAO DE VALORES DOS EIXOS Y E Z
 else if(arrayLocomocao[vtOrdenada] <= -7 && arrayLocomocao[vtCota] >= 0){     //PARA ANDAR PARA TRAS E NECESSARIO TER VALORES EM -Y E Z
```

```
   BrunuhSGS.send("Andar Para Tras");                //ENVIA UMA MENSAGEM PARA O AMARINO, INFORMANDO QUAL O PROCEDIMENTO A SER REALIZADO
   noInterrupts();
   locomocaoFrente();        //CHAMA A FUNCAO QUE ANDA PARA FRENTE
  }
  //ANDAR PARA ESQUERDA: COMPARACAO DE VALORES DOS EIXOS -X E Z
  else if(arrayLocomocao[vtAbscissa] <= -2 && arrayLocomocao[vtCota] >= 4){     //PARA ANDAR PARA ESQUERDA E NECESSARIO TER VALORES EM -X E Z
   BrunuhSGS.send("Andar Para Esquerda");           //ENVIA UMA MENSAGEM PARA O AMARINO, INFORMANDO QUAL O PROCEDIMENTO A SER REALIZADO
   noInterrupts();
   locomocaoDireita();        //CHAMA A FUNCAO QUE ANDA PARA DIREITA
  }
  //ANDAR PARA DIREITA: COMPARACAO DE VALORES DOS EIXOS X E Z
  else if(arrayLocomocao[vtAbscissa] >= 2 && arrayLocomocao[vtCota] >= 4){      //PARA ANDAR PARA DIREITA E NECESSARIO TER VALORES EM X E Z
   BrunuhSGS.send("Andar Para Direita");            //ENVIA UMA MENSAGEM PARA O AMARINO, INFORMANDO QUAL O PROCEDIMENTO A SER REALIZADO
   noInterrupts();
   locomocaoEsquerda();        //CHAMA A FUNCAO QUE ANDA PARA ESQUERDA
  }*/
  else{
   BrunuhSGS.send("Nenhuma Funcao Ativa");    //INFORMA O PROTOTIPO NAO REALIZOU NENHUMA ATIVIDADE
   noInterrupts();
   prototipoFreioVT();           //CHAMA A FUNCAO QUE PARA O PROTOTIPO CASO NENHUMA FUNCIONALIDADE ESTEJA ATIVA
  }
}

//FUNCAO PARA O PROTOTIPO SE LOCOMOVER PARA FRENTE
void locomocaoFrente(){
  analogWrite(servoLeft, ctFrente);    //SERVO MOTOR DA ESQUERDA ANDAR PARA FRENTE
  analogWrite(servoRight, ctTras);     //SERVO MOTOR DA DIREITA ANDAR PARA TRAS
  interrupts();
}

//FUNCAO PARA O PROTOTIPO SE LOCOMOVER PARA TRAS
void locomocaoTras(){
  analogWrite(servoLeft, ctTras);      //SERVOMOTOR DA ESQUERDA ANDAR PARA TRAS
  analogWrite(servoRight, ctFrente);   //SERVOMOTOR DA DIREITA ANDAR PARA FRENTE
```

```
  interrupts();
}

//FUNCAO PARA O PROTOTIPO ABRIR O VIDRO
void vidroAbrir(){
  analogWrite(servoVidro, 10);        //SERVOMOTOR DO VIDRO PARA ABRIR
  digitalWrite(buzina, HIGH);
  digitalWrite(farol, HIGH);
  //digitalWrite(farolRight, HIGH);
  delay(10);
  digitalWrite(buzina, LOW);
  digitalWrite(farol, LOW);
  //digitalWrite(farolRight, LOW);
  interrupts();
}

//FUNCAO PARA O PROTOTIPO FECHAR O VIDRO
void vidroFechar(){
  analogWrite(servoVidro, 240);        //SERVOMOTOR DO VIDRO PARA FECHAR
  interrupts();
}

//FUNCAO PARA O PROTOTIPO ABRIR A TRAVA
void travaAbrir(){
  analogWrite(servoTrava, 10);        //SERVOMOTOR DO VIDRO PARA ABRIR
  interrupts();
}

//FUNCAO PARA O PROTOTIPO FECHAR A TRAVA
void travaFechar(){
  analogWrite(servoTrava, 210);        //SERVOMOTOR DO VIDRO PARA FECHAR
  interrupts();
}

/*
//FUNCAO PARA O PROTOTIPO SOAR A BUZINA
void buzinaAtiva(){
  if(valorStr == 'A' || valorStr == 'C'){
    digitalWrite(buzina, 100);
    delay(10);
    digitalWrite(buzina, LOW);
```

```
  }
  else if(valorStr == 'B' || valorStr == 'D'){
    digitalWrite(buzina, 200);
    delay(10);
    digitalWrite(buzina, LOW);
  }
  interrupts();
}*/

//FUNCAO PARA OS MOTORES DA ESQUERDA E DIREITA
void prototipoFreioVT(){
  analogWrite(servoLeft, 0);     //SERVOMOTOR DA ESQUERDA FICA PARADO
  analogWrite(servoRight, 0);    //SERVOMOTOR DA DIREITA FICA PARADO
  interrupts();
}
```