CENTRO UNIVERSITÁRIO DE BRASÍLIA – UniCEUB

FATECS – FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS

CURSO DE ENGENHARIA DE COMPUTAÇÃO

MARIANA PATRÍCIA PEREIRA DE SOUZA

ACIONAMENTO DE MOTOR ELÉTRICO VIA BLUETOOTH DE CELULAR

Orientadora: M.C. Maria Marony Sousa Farias

Brasília

Junho, 2010

MARIANA PATRÍCIA PEREIRA DE SOUZA

ACIONAMENTO DE MOTOR ELÉTRICO VIA BLUETOOTH DE CELULAR

Trabalho apresentado ao Centro Universitário de Brasília como pré-requisito para a obtenção de Certificado de Conclusão do Curso de Engenharia de Computação.

Orientadora: M.C. Maria Marony Sousa Farias

Brasília

Junho, 2010

MARIANA PATRÍCIA PEREIRA DE SOUZA

ACIONAMENTO DE MOTOR ELÉTRICO VIA BLUETOOTH DE CELULAR

Trabalho apresentado ao Centro Universitário de Brasília como pré-requisito para a obtenção de Certificado de Conclusão do 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. Abiezer Amarilia Fernandez
Coordenador do Curso

Banca Examinadora:

_____
Profª. Maria Marony Sousa Farias, mestre em Engenharia Elétrica – UFPB – PB.
Orientadora

_____
Prof. Antonio Barbosa Junior
UniCEUB

_____
Prof. MSC Francisco Javier de Obaldia Diaz, mestre em Engenharia Elétrica
UniCEUB

AGRADECIMENTOS

Primeiramente agradeço a Deus.

Gostaria de agradecer a minha orientadora por todo apoio. Também gostaria de agradecer aos meus amigos Lucas e Malu que sempre acreditaram e estiveram presentes durante no desenvolvimento deste projeto.

Um agradecimento especial à minha família, a minha tia Betânia e seu marido Jefferson, pelo apoio incondicional durante o desenvolvimento deste projeto.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

A – Ampère, unidade de medida de intensidade de corrente elétrica

AC – Alternating Current

AT – Hayes AT Commands

bps – Bit per second, unidade de transmissão de dados de bits por segundo

CA – Corrente Alternada

CC – Corrente Contínua

CI – Circuito Integrado

CM - centímetros

CPU – Central Processing Unit

DC – Direct Current

DVD-ROM – Compact Disc Read-Only Memory

Endereços MAC – Media Access Control

GND – Ground, terra, aterramento

HEX – Hexadecimal

I/O – Input/Output

IBGE – Instituto Brasileiro de Geografia e Estatística

IDE – Integrated Development Environment

ISO - International Organization of Standardization

kbit/s – Kilobit por segundo, unidade de transmissão de dados de 1000 bits por segundo

kbps – Kilobit por segundo, unidade de transmissão de dados de 1000 bits por segundo

L2CAP – Logical Link Control and Adaptation Protocol

LC - Link Controller

LED – Light Emitting Diode

LM – Link Manager

LMP- Link Manager Protocol

MCU – Microcontrolador

MHz – Mega ou 10^6 hertz, unidade de frequência

NA – Normal Aberto

NF – Normal Fechado

OBEX - Object Exchange

OSI - Open System Interconnection

PPP - Point-to-Point Protocol

PWM – Pulse Width Modulation

RAM – Random Access Memory

RF - rádio-frequência

ROM – Read-Only Memory

rpm – Rotações por minuto, é uma unidade de velocidade angular

RS232 – Recommended Standard 232

SDP - Service Discovery Protocol

SIG - Bluetooth Special Interest Group

TCP/IP – Transmission Control Protocol/Internet Protocol

TCS-BIN - Telephony Control Specification – Binary

TDD - Time Division Duplex

TTL – Transistor-Transistor Logic

UART – Universal Asynchronous Receiver Transmitter

UCP – Unidade Central de Processamento

UDP - User Datagram Protocol

USART – Universal Syncronous Asyncronous Receiver Transmitter

USB – Universal Serial Bus

V – Volt, unidade de tensão elétrica

VAC – Volt Alternating Current

VDC – Voltage Direct Current

W – Watt, unidade de potência

WAE - Wireless Application Environment

WAP - Wireless Application Protocol

Wi-Fi - Wireless Fidelity

μ – Micro, um fator de 10^{-6}, unidade de milionésimo

Ω – Ohm, unidade de resistência

RESUMO

Neste trabalho é apresentada uma aplicação prática para o acionamento de um motor elétrico via Bluetooth de celular por meio de um protótipo. No projeto apresentado, o motor é acionado através de uma placa Arduino. Depois do reconhecimento do dispositivo Bluetooth autorizado. Uma vez acionado, a porta deslizante é aberta e os sensores fim de curso fazem a leitura do posicionamento da mesma. Após a leitura dos sensores, o LED, estrategicamente localizado indica o estado do portão: a cor vermelha sinaliza que o portão está fechado, verde indica que está aberto e azul indica que o portão está em movimento. Por fim, para a movimentação da porta deslizante é necessário utilizar um motor DC controlado por um microcontrolador, presente na placa Arduino Mega.

Palavras Chave: Arduino Mega, motor DC, módulo Bluetooth, sensores, e autorização de dispositivos.

ABSTRACT

This project presents a practical application designed to enable an electric motor via Bluetooth of a cell phone through a prototype. In this project, the motor is enabLED through the Arduino board and a Bluetooth module, after the authentication of a pre-registered Bluetooth device. Once initialize, the sliding door opens and the *reed-switch* sensors reads the position in which the sliding door is currently. After the reading of the sensors, the strategically located LED indicates the status of the gate. The red color indicates that the gate is closed, the green indicates that it is open and blue indicates that the gate is in movement. Thus, in order to move the sliding door, it is necessary to use a DC motor controlled by a microcontroller, present in an Arduino Mega board.

Keywords: Arduino Mega, DC Motor, Bluetooth module, sensors, and authentication of devices.

CAPÍTULO 1 - INTRODUÇÃO

O tema de automação de rotinas do cotidiano não é tão recente. Com os novos aplicativos de celular, mais pessoas procuram consumir tudo a sua volta. Segundo o filósofo francês Gilles Lipovetsky, o consumo é um dos raros fenômenos que conseguiram modificar profundamente os modos de vida, os gostos, as aspirações e os comportamentos da maioria, em um intervalo de tempo tão curto. (IBOPE - 2009). Consumir, do verbo transitivo usar, utilizar, é o retrato do consumidor de hoje. Com isso em mente, o tema proposto se torna uma aplicação útil e prática para o dia a dia dos portadores de celulares com Bluetooth.

Hoje em dia, vivemos em um mundo que está se movimentando globalmente para a integração e automação de produtos. Fazemos parte de uma sociedade que utiliza o computador e dispositivos do gênero em diferentes seções de nossas vidas. Quando vamos ao supermercado, ao shopping, ao banco e até mesmo ao trabalho nos deparamos com automação e controle de dispositivos. Seguindo essa tendência, o próximo setor afetado é o residêncial. Já utilizamos dispositivos eletrônicos como o motor elétrico da garagem de uma residência ou condomínio. Grande parte desses motores são acionados por um controle de infravermelho. No entanto, esse tipo de acionamento não possui um controle de usuários preciso e também é realizado por meio de outro dispositivo, que poder ser extraviado além de ser um item há mais a ser carregado pelo usuário. Pesquisas mostram que entre as necessidades que determinam ritmo às mudanças no padrão de consumo, o celular tem papel fundamental. Na lista de itens mais importantes no dia a dia o telefone celular figura com 70% de prioridade. (PRESS - 2009) A revolução dos novos dispositivos móveis e seus infinitos aplicativos justificam a "multiplicação indefinida das necessidades" dos mesmos. (LIPOVETSKY – 2008). O que justifica o interesse em utilizar aplicações já existentes na tecnologia de dispositivos móveis para prover mais comodidade aos usuários.

1.1– Motivação

Conforto e praticidade são objetivos dos usuários de tecnologia de hoje dia. Com isso em mente, a proposta deste projeto é aumentar opções de comodidade dos usuários de dispositivos móveis aproveitando aplicações já existentes nos mesmos. O desenvolvimento de uma aplicação para simbolizar o acionamento de um motor elétrico de garagem, por exemplo,

contribuem para concretização deste projeto, que tem como objetivo gerar mais praticidade para os usuários de celulares.

A motivação para a realização desde projeto surgiu a partir da observação da variedade de conexões sem fio presentes nos dispositivos móveis. Dispositivos móveis, como celulares, notebooks e IPads, possuem conexões sem fio Wi-Fi e Bluetooth. Wi-Fi sempre é utilizado para a navegação da internet. No entanto, poucos utilizam a conexão Bluetooth. E quando a utilizam, geralmente é para a transferência de arquivos.

Com a ideia de ampliar a utilização dos dispositivos móveis, este projeto propõe, por meio de um protótipo, à construção de um portão para a demonstração do acionamento de um motor elétrico. Para tal demonstração, um celular se comunicará via Bluetooth com o módulo Bluetooth em um Arduino pré-programado. O Arduino por sua vez receberá a informação do celular do usuário, previamente cadastrado, e verificará se o usuário está ou não autorizado a entrar. Após a identificação do usuário autorizado, o Arduino enviará sinais de controle a um circuito eletrônico para acionar o motor DC. O acionamento do motor resultará na abertura da porta deslizante, o portão. O *status* do motor será identificado por dois sensores estrategicamente instalados. Os sensores localizam-se na extremidade final e inicial da parede, e o outro na extremidade final do portão para identificar a abertura e o fechamento total do portão. A leitura desses sensores resultará na iluminação do LED RGB vermelho, verde ou azul para sinalizar o estado do portão. Com o portão fechado e o motor parado, o LED permanecerá vermelho. Assim que o motor é acionado e o portão começar a abrir, o LED ficará azul até a identificação do sensor na extremidade final da parede, o que assim iluminará verde. Após treze segundos da abertura completa do portão, o motor é acionado novamente para realizar o fechamento do portão.

O projeto restringe a demonstrar esta solução em forma de protótipo. No entanto, pode ser adequado à realidade. Contudo, neste protótipo são utilizados materiais de escala reduzida, como por exemplo, o protótipo do portão e o motor DC com redução para simular a abertura e o fechamento do portão. Adequando o projeto a realidade, o portão utilizaria motores mais potentes.

1.2 – Visão Geral do Projeto

O projeto simula o acionamento de um motor elétrico em um portão deslizante. Este protótipo define que o motor seja acionado mediante a autenticação do dispositivo Bluetooth. Logo após o acionamento do mesmo, o usuário do celular pode verificar o estado do portão, por meio de um LED, sem ter que estar diante dele. As cores projetadas pelo LED possibilitam uma maior interação e comodidade do usuário para saber se o portão está aberto, fechado ou travado. Dessa maneira o usuário pode então tomar a providência necessária para o caso de algum problema.

Foi utilizado um módulo Bluetooth acoplado em uma placa Arduino Mega para estabelecer a identificação do Bluetooth do dispositivo móvel, como o celular. A placa Arduino Mega, que foi programada, também foi utilizada para verificar os dispositivos Bluetooth presentes na área, assim como também identificar os dispositivos autorizados. Uma vez identificados e autorizados, o Arduino aciona motor DC, fazendo com que o portão abra e feche de acordo com a programação.

Para poder interpretar os comandos de abertura e fechamento do portão e transformá-los em sinais capazes de acionar o motor DC, foi utilizado um circuito integrado, CI, L293D. Esse CI também é responsável por controlar a velocidade de rotação do motor.

Sensores de fim de curso e um LED RGB foram utilizados para demonstrar o estado no qual o portão se encontra. A leitura dos sensores e o acionamento do LED também são feitos pela Arduino Mega, de acordo sua programação.

O quadro 1.1 ilustra os dispositivos eletrônicos utilizados na montagem do protótipo do projeto.


Protótipo do portão


Celular

Arduino Mega


Módulo Bluetooth


MegaShield

Erro! Indicador não definido.

BreadBoard

L293D


Chave fim de curso


LED RGB


Motor DC reduzido

Quadro 1.1 – Dispositivos eletrônicos utilizados no projeto.

1.3 – Objetivos

A finalidade deste projeto é demonstrar uma aplicação prática para o uso do Bluetooth de um celular por meio de um protótipo de um portão utilizando um motor DC. Aumentando assim, as opções de aplicações já existentes na tecnologia de dispositivos móveis. Resultando então na comodidade dos usuários desses dispositivos. O desenvolvimento de uma aplicação para simbolizar o acionamento de um motor elétrico de garagem contribui para concretização deste projeto que tem como objetivo gerar mais praticidade para os usuários de celulares.

Assim, a construção de um protótipo de um portão relatada neste projeto proporciona uma opção prática e simples para sua utilização. Com isso, favorece então o aumento da comodidade do usuário.

O objetivo geral deste trabalho é apresentar um protótipo de um portão utilizando um motor DC. Este portão abrirá conforme o reconhecimento do dispositivo autorizado. O funcionamento em conjunto com o portão, os sensores de fim de curso detectarão a posição do portão. Com isso, o LED então ilumina com a cor correspondente a posição do mesmo. O

controle da parte lógica do projeto é de responsabilidade da placa Arduino Mega, contendo toda a programação do funcionamento gravada em sua memória flash.

Para o completo funcionamento deste projeto, as tarefas abaixo foram executadas:

- Construir um protótipo simulando um portão;
- Utilizar um Arduino Mega para gerenciar a parte lógica do projeto;
- Criar um programa utilizando Arduino IDE para controlar todo o sistema do portão e controle de dispositivos;
- Utilizar um motor DC reduzido para abrir e fechar o portão;
- Utilizar CI L293D com circuito de ponte H interno para facilitar o funcionamento do motor DC;
- Utilizar um módulo Bluetooth para reconhecer os dispositivos Bluetooths ativados na área;
- Utilizar um MegaShield para transpor os pinos do Arduino Mega para possibilitar montagem do circuito do motor e a comunicação com o módulo Bluetooth com Arduino;
- Utilizar sensores de fim de curso para informar a posição atual do portão, aberto, fechado ou travado;
- Utilizar LEDs para a indicação do *status* do portão, na cor vermelha quando fechado, verde quando aberto, azul quando em movimento e piscando azul quando travado;

1.4 – Definição do Problema

Mencionado previamente, o problema estudado e solucionado neste projeto teve como objetivo propor uma opção de utilidade à conexão sem fio Bluetooth no cotidiano. Em Brasília, por exemplo, existem várias residências em condomínios fechados, chácaras, fazendas e colônias agrícolas, cujo os donos poderiam usar o Bluetooth para abrir os portões de tais locais, facilitando o acesso, aumentando a segurança. Os dispositivos teriam que ser cadastrados previamente e os moradores ou frequentadores dos mesmos teriam mais comodidade já que não teriam que carregar um controle especificamente para abrir o portão.

Dentre os diversos tipos de controle para acionar portões presentes hoje em dia, o acionamento de motor elétrico via Bluetooth foi a proposta para a resolução do problema

relatado neste trabalho. Explorar as utilidades das tecnologias já presentes nos dispositivos móveis favoreceu a comodidade do usuário.

1.5 – Restrições

A principal vantagem deste experimento é não utilizar ou carregar controles para acionar o motor. Contudo, o sistema criado neste trabalho se limita a isso, não considerando assim outros aspectos de práticas e segurança que possam vir a ser questionados.

O escopo do projeto não contempla o controle do motor através do dispositivo móvel, celular. Somente o Arduino possui esse tipo de controle. Dessa forma, o protótipo não permite a programação e a configuração do Arduino de forma remota pelo celular e/ou pelo dispositivo móvel.

Já para a implementação do sistema desenvolvido, tem-se a necessidade de conectar o motor a uma fonte de energia e que o dispositivo móvel tenha bateria suficiente para estar ativado. Todavia, este projeto não contempla recursos que possibilitem mantê-lo ligado após falta de energia, como *no-break* ou outras fontes de alimentação, ou até mesmo problemas no motor ou no dispositivo móvel.

Questões de segurança não se aplicam ao projeto. A única questão de segurança apresentada refere-se ao cadastro dos endereços MAC dos dispositivos móveis no Arduino. A interferência do deslizamento do portão por obstáculos exteriores, também não são tratados. Podemos citar como exemplo a parada do carro entre o portão. Assim sendo, se o portão for acionado, o motor tentará fechá-lo ininterruptamente até que o processo se complete. É nesse momento que o Arduino verifica os estados das chaves fim de curso.

Outra questão de segurança não abordada é a possibilidade do acionamento do motor antes da entrada do motorista no carro, na quadra ou até mesmo em sua permanência em sua residência. Sabendo que o Bluetooth tem um alcance de 10 metros, sugere-se ativar o Bluetooth somente na saída da residência e desativá-lo após sua saída. Ou em sua entrada. Dessa maneira, o motor não será acionado indevidamente. É importante lembrar que o objetivo deste projeto é propor mais utilidade para a conexão Bluetooth.

Similarmente, questões de intrusão de pessoa desconhecida não autorizada, no caso do usuário simplesmente passar próximo do portão ou acionar sem querer do motor, também não

fazem parte do escopo deste projeto. Em casos assim, outros itens de segurança devem ser acoplados no sistema do portão, como diminuir a distância de abrangência do sinal Bluetooth para aquela aplicação ou um segurança próximo dali. Porém, se esse sistema de acionamento for implantado em edifícios, a ativação e desativação da conexão Bluetooth terá que ser feita pelo próprio usuário de acordo com a necessidade do mesmo e de seu apartamento.

O protótipo desenvolvido para o portão é de baixa escala. Com isso, em casos de travamento, o conserto deve ser manual. A utilização do LED para avisar o usuário de qualquer problema antes mesmo da chegada à frente do portão, como em casos de utilização em edifícios, foi implementado. Destacando-se na proposta do sistema, o aviso luminoso do mesmo alertando o usuário se o portão abriu ou não por completo.

É importante ressaltar também que não é o objetivo deste projeto desenvolver nenhum aplicativo e nenhuma interface interativa para usuário cadastrar seu dispositivo. Assim como também, não faz parte do escopo o desenvolvimento de dispositivos tais como um conversor Bluetooth para infravermelho (BT2IR), o acionamento do motor diretamente de um dispositivo móvel, ou o recebimento do *status* do portão diretamente no dispositivo móvel.

1.6 – Diagrama de Blocos

A figura 1.1 ilustra o diagrama de blocos deste projeto onde pode se observar os principais componentes eletrônicos e o processo de funcionamento do protótipo do projeto.

O diagrama ilustra o controle de um portão deslizante com um módulo Bluetooth e Arduino, que verificam os dispositivos Bluetooth ativados e autorizados. O programa desenvolvido foi gravado na memória flash do Arduino, de forma que permite o mesmo controlar toda parte lógica do projeto.

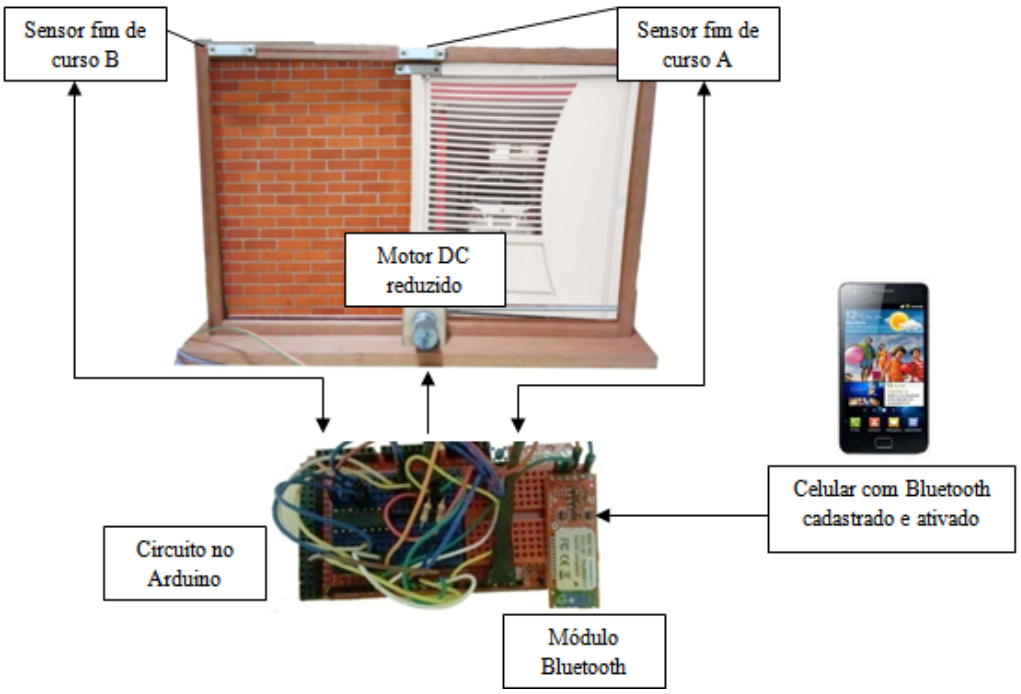


Figura 1.1 – Diagrama de blocos do projeto.

1.7 – Metodologias

Consultas a artigos científicos, sites de internet e livros foram utilizadas como fontes bibliográficas para a realização da montagem do protótipo. Similarmente, estudos dos componentes, dispositivos eletrônicos, a linguagem de programação, IDE, foram realizados para a compreensão do desenvolvimento do sistema. Após a compreensão desses dispositivos, testes foram realizados pra a confirmação de seu funcionamento na implementação.

O principal componente para garantir o correto funcionamento de todo o sistema foi o Arduino Mega. A placa Arduino permite fazer a comunicação entre todos os dispositivos mecânicos e eletrônicos de tal forma a atingir os objetivos propostos. Além do Arduino, outro elemento chave deste projeto é o módulo Bluetooth Mate, que por sua vez, reconhece os dispositivos Bluetooth ativados de uma área.

1.8 – Estrutura da Monografia

Esta monografia é composta de 6 capítulos. Como primeiro capítulo, a INTRODUÇÃO, apresentará a motivação do projeto, suas restrições, os principais objetivos, a metodologia de pesquisa, o diagrama de blocos, assim como também toda estrutura desse trabalho.

No capítulo 2 são apresentados os FUNDAMENTOS TEÓRICOS, com descrição dos principais assuntos abordados fazendo então menção aos conceitos de redes sem fios, Bluetooth, motor elétrico DC, o detalhamento dos sensores fim de curso, Arduino e suas vantagens ligadas a este projeto,

No capítulo 3, a DESCRIÇÃO DO HARDWARE utilizado no projeto. Neste capítulo é detalhada a especificação da placa Arduino Mega, configuração da mesma, do módulo Bluetooth Mate assim como o do motor DC.

Já no capítulo 4 é apresentado todo detalhamento da IMPLEMENTAÇÃO da proposta do projeto. A topologia, os circuitos construídos e o código fonte desenvolvido para a solução do problema são explicados no mesmo.

No capítulo 5 são apresentados os TESTES E RESULTADOS da solução proposta para o problema. A apresentação do circuito completo na placa final e o protótipo concluído também são apresentados aqui.

O capítulo 6 apresenta a CONCLUSÃO. Este capítulo contém o encerramento da monografia junto com as dificuldades encontradas. Este capítulo também inclui propostas para trabalhos futuros, bem como sugestões para o prosseguimento de trabalhos a fins.

CAPÍTULO 2 – FUNDAMENTOS TEÓRICOS

Para melhor compreensão do projeto e seu desenvolvimento, são necessários alguns conhecimentos quanto a teoria. Neste capítulo são apresentados os fundamentos teóricos para o correto entendimento do projeto tais como: redes sem fios, tipos de acionamento de motores elétricos no mercado atual, Bluetooth, motor elétrico e Arduino.

2.1 –Redes Sem fio

Segundo (SOUSA – 2002) uma rede sem fio, ou rede Wireless, se refere a uma rede que não requer o uso de cabos e utiliza sinais de radio-frequência para a transmissão de dados à uma velocidade de 11Mbps. Apesar de serem ideais para o uso ao ar livre, também são bem úteis em ambientes fechados que possuem limitações físicas para a passagem de cabos assim como orçamentárias. Nesse tipo de sistema de comunicação, a transmissão é feita através do ar, ou até mesmo o vácuo, utilizando canais de frequência de rádio ou a tecnologia de espectro espalhado, como o infravermelho por exemplo. As redes sem fios baseadas em radiodifusão, por serem mais utilizadas em redes de computadores, segundo (SOARES – 1995), utilizam o padrão IEEE (*Institute of Eletronic and Electrical Enginers*) 802.11. Esse padrão suporta redes de computadores dentro de um raio de até 150 metros de distância. O quadro 2.1 ilustra o desempenho dos diferentes tipos de rede.

Existem diversos tipos aplicações para redes sem fio e o tipo de aplicação a ser abordada neste trabalho é a rede móvel e portátil, a da telefonia celular. Essas redes fazem parte das redes pessoais denominadas de WPANs (*Wireless Personal Area Network*). (TANENBAUM -2003).

	Exemplo	*Alcance*	*Largura de banda (Mbps)*	*Latência (ms)*
Redes cabeadas:				
LAN	Ethernet	1 - 2 kms	10 - 1000	1 - 10
WAN	Roteamento IP	mundial	0,010 - 600	100 - 500
MAN	ATM	2 - 50 kms	1 - 150	10
Interligação em rede	Internet	mundial	0.5 - 600	100 - 500
Redes sem fio:				
WPAN	Bluetooth (IEEE 802.15.1)	10 - 30 kms	0,5 - 2	5 - 20
WLAN	WiFi (IEEE 802.11)	0,15 - 1,5 kms	2 - 54	5 - 20
WMAN	WiMAX (IEEE 802.16)	5 - 50 kms	1,5 - 20	5 - 20
WWAN	Redes telefônicas GSM 3G	mundial	0,010 - 2	100 - 500

Quadro 2.1 – Desempenho de Rede

(Fonte: COLOURIS - 2007, Página 74)

2.1.1 – Bluetooth

A tecnologia Bluetooth é uma tecnologia de comunicação sem fio destinada a substituir os cabos de ligação portátil assim como também dispositivos fixos, mantendo certos níveis de segurança. Segundo (SIG – 2010) as principais características da tecnologia Bluetooth são a robustez, baixo consumo de energia e baixo custo. A especificação Bluetooth possui estrutura uniforme que possibilita a conectividade e a comunicação de uma ampla diversificação de dispositivos como celulares, fone de ouvido, microfones, teclado, impressoras e até mesmo computadores sem a necessidade de conectar outro dispositivo.

Em homenagem ao rei dinamarquês Harald Bluetooth, essa tecnologia unificadora recebeu o nome de Bluetooth por unificar, de uma maneira prática, outros dispositivos. Essa tecnologia de conexão sem fio (wireless) a curta distância opera em uma banda de 2,45 GHz denominada ISM (*Industrial, Scientic and Medical*) que funciona em uma distância de até 10 metros. (MILLER – 2001, 9) Os sinais de rádio-frequência (RF) da tecnologia Bluetooth consegue atravessar certos objetos sólidos o que amplia sua gama de dispositivos que podem utilizá-la.

Um dispositivo Bluetooth possui um transceptor (*tranceiver*) de curto alcance que por esse necessita menos potência e resulta em um consumo menor de energia facilitando sua empregabilidade. O Bluetooth faz parte de uma rede *Ad Hoc.* A expressão Ad Hoc vem da expressão do latim que significa “para esta finalidade” ou “com este objetivo”. Neste caso, uma rede Ad Hoc é aquela que possui uma arquitetura que permite se comunicar apenas com a aproximação dos dispositivos. No entanto com o distanciamento dos dispositivos a rede é

desfeita sem o conhecimento ou comando do usuário, dando a essa tecnologia uma característica de automação. Com isso, as redes Bluetooth têm como característica s formação de grupos de transmissão denominados de piconets. Piconet é um conjunto de até oito dispositivos Bluetooth. Um dispositivo é designado como mestre e os outros são como escravos. Mais detalhes sobre piconet será tratado no tópico de segurança em redes Bluetooth.

O núcleo do sistema Bluetooth é composto por um transceptor de RF, controle de banda (*baseband)* e uma pilha de protocolo. Com essa estrutura, esse sistema oferece serviços que permitem a conexão de dispositivos e o intercâmbio de uma variedade de dados entre esses dispositivos. A figura 2.1 abaixo ilustra o controlador Bluetooth. (SIG – 2010) (MILLER – 2001)

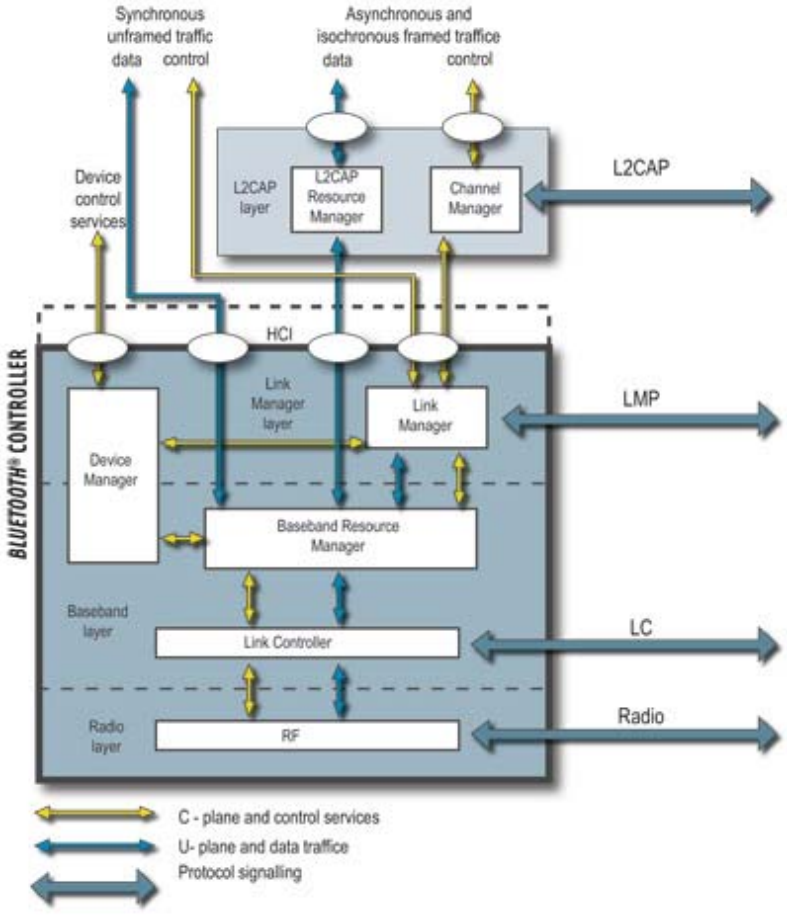


Figura 2.1 – Ilustração do Controlador Bluetooth

(FONTE: https://www.bluetooth.org/Building/HowTechnologyWorks/Architecture/Overview.htm Acesso em 10 de maio de 2009)

2.1.2 – Pilha de protocolos

O padrão Bluetooth foi baseado no modelo *Open System Interconnection* (OSI). O Modelo OSI é um modelo que possui sete camadas definidas pela *International Organization of Standardization* (ISO). Em outras palavras, esse modelo é uma forma de sub-dividir um sistema em partes pequenas denominados de camadas, onde cada camada é uma coleção de funções similares que prestam serviços à camada acima dela e recebe serviços da camada abaixo dela. A padronização deste permite a comunicação entre computadores e dispositivos de diferentes fabricantes. Os principais protocolos Bluetooth fornecem funções de transporte e gerenciamento de link a todas as aplicações. A figura 2.2 ilustra a organização dessa pilha de protocolos.

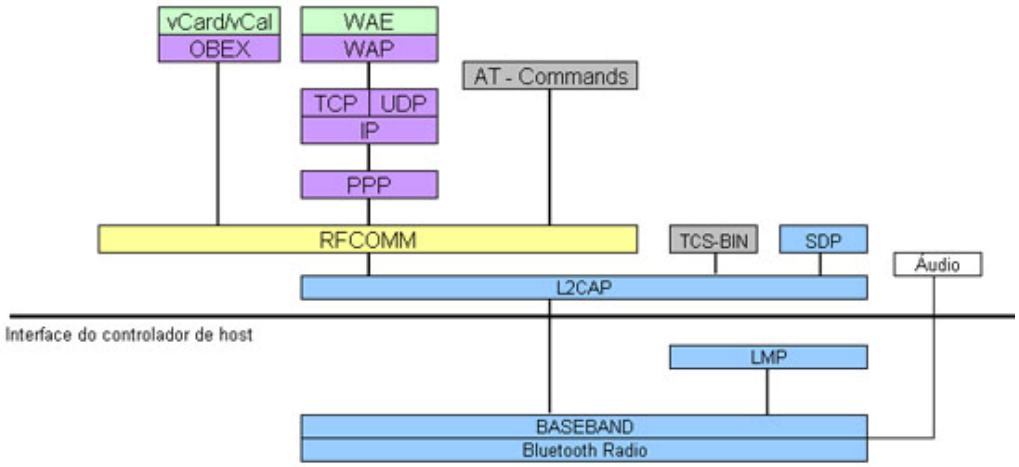


Figura 2.2 – Pilha de Protocolos Bluetooth

(FONTE: MILLER – 2001, Página 152)

Observando o diagrama da pilha de protocolos vemos a divisão da pilha onde a parte inferior denominada de interface do controlador de host, que é designado à camada física dos protocolos responsável pela manutenção dos enlaces com as camadas *Bluetooth Radio, Baseband* e *Link Manager Protocol* - (*LMP*), e a superior designada camada de aplicação. (MILLER – 2001)

Os protocolos *Bluetooth Radio, Baseband* e *Link Manager Protocol* (*LMP*) formam um grupo denominado de *Core Protocol*, ou simplesmente protocolos centrais. Um dispositivo só pode ser caracterizado como Bluetooth se tiver o *Core Protocol.* A camada Bluetooth Radio, assim como a Baseband, equivale a camada física do modelo OSI. É nessa camada que é analisado a transmissão RF e sua modulação. Segundo (SIG – 2010), bloco de RF é responsável por transmitir e receber pacotes de informações sobre o canal físico. O canal entre o controle de banda e do bloco de RF permite que o bloco de baseband possa controlar o tempo e a frequência portadora do bloco de RF. Com isso, o bloco RF transforma o fluxo de dados de e para a camada física e baseband nos formatos exigidos. Já a camada Baseband é onde se encontra a especificação do Controlador de Enlace (*Link Controller-LC*) do Bluetooth. O controlador de enlace é responsável pela codificação e decodificação dos pacotes Bluetooth a partir do bloco de dados e parâmetros relacionados com a camada física, transporte e lógica. Ele é utilizado para comunicar o controle de fluxo e sinais de reconhecimento e até mesmo o pedido de retransmissão do dado. Por sua vez o protocolo LMP (Protocolo de Gerenciamento de Enlace) é responsável pelo controle dos enlaces assim como a configuração dos mesmos e é usado pelo Gerenciador de Enlace (*LM – Link Manager*). O *Link Manager* é responsável pela criação, modificação e liberação de ligações lógicas.

A camada de aplicação engloba todos os blocos acima da interface do controlador de host. O protocolo de Adaptação e controle de Enlace Lógico (L2CAP – *Logical Link Control and Adaptation Protocol*) faz a segmentação e a montagem dos pacotes assim como a multiplexação e de multiplexação dos pacotes permitindo que um dispositivo mantenha mais do que uma conexão com diferentes dispositivos simultaneamente. Esse protocolo também é descrito como protocolo de acesso ao meio. O protocolo acima da camada L2CAP, RFCOMM, faz parte do protocolo *Cable Replacement* (Substituição de cabo) e é utilizado para simular uma comunicação serial entre dois dispositivos fornecendo recursos de transporte para os serviços de nível superior. (MILLER – 2001)

O protocolo *Telephony Control Specification – Binary (TCS-BIN)* foi desenvolvido pelo Bluetooth SIG e responsável por definir a sinalização de controle de camada necessária para estabelecer chamadas de voz e dados entre os dispositivos Bluetooth. Por sua vez, o protocolo de Descoberta de Serviços, *Service Discovery Protocol (SDP)*, provê meios para que a aplicação possa descobrir quais serviços estão disponíveis nos dispositivos Bluetooth. Ele permite que dois dispositivos Bluetooth diferentes possam estabelecer uma conexão e se

comunicarem. O comando AT (*AT Command*) é um conjunto de comandos de áudio e de telefonia que permite controlar funções desempenhadas por um telefone ou um modem de dados. (MILLER – 2001)

Buscando maior compatibilidade com outros padrões, os protocolos OBEX, WAP, UDP, TCP, IP e PPP são adotados pelo Bluetooth. Conhecido como protocolo PPP, o *Point-to-Point Protocol*, é o protocolo que define como os dados do *Internet Protocol* (IP) serão transmitidos através dos links serias de ponto-a-ponto. Esse protocolo estabelece um método de acesso a Internet por meio de um computador ligado a um host de Internet por um telefone ou a um modem com alta velocidade. (MILLER – 2001)

A tecnologia Bluetooth adotou os protocolos TCP, IP e UDP para facilitar a comunicação de seus dispositivos com qualquer outro dispositivo conectado a Internet, já que são protocolos tradicionais de conexões de Internet. O protocolo TCP, *Transport Control Protocol*, delimita o procedimento que divide os dados em pacotes e que após a transmissão os reúne no outro lado. Já o protocolo IP define como os dados serão enviados pelos roteadores para redes diferentes atribuindo um endereço único chamado de IP. For fim, o protocolo UDP (*User Datagram Protocol*) é utilizado para transmitir mensagens individuais para o IP sem a confirmação de entrega da mesma. (MILLER – 2001)

O protocolo *Object Exchange (OBEX)*, é utilizado para facilitar o intercâmbio de objetos ou dados entre dispositivos diferentes. Esse intercâmbio de informação dispõe-se de uma arquitetura cliente-servidor que permite examinar as pastas armazenadas nos dispositivos remotos em formatos *vCard, vCalendar, vMessage* e *vNote*.

Ainda implementando serviços de internet em dispositivos móveis, o protocolo de aplicação sem fio, *Wireless Application Protocol (WAP)*, capacita o telefone móvel, celular, a navegar na internet. Com isso, a tecnologia Bluetooth possui sua própria pilha de protocolos WAP especifica como está ilustrado na figura 2.3.

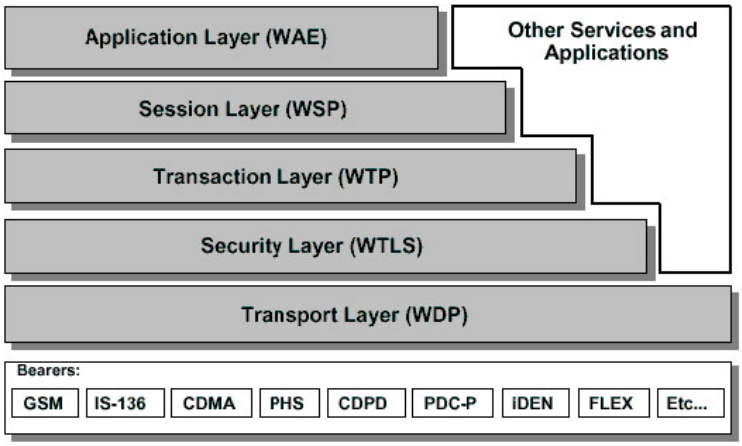


Figura 2.3 – Pilha de Protocolos WAP

(FONTE: CEFETRIO, 2010)

O protocolo *Wireless Application Environment (WAE)* é a camada da pilha WAP, ilustrada acima, fornece um ambiente interativo onde as operadoras e prestadores de serviços constroem aplicações para as plataformas sem fio. Ele também favorece os softwares designados para a utilização em celulares assim como também dispositivos compatíveis com Bluetooth. (SIG – 2010) (MILLER – 2001)

2.1.3 – Comunicação e Segurança em redes Bluetooth

Como mencionado anteriormente, os dispositivos Bluetooth se comunicam através de uma rede chamada de piconet. Um piconet é dispositivo de uma área que seguem uma mesma regra de sincronização onde o tempo é dividido em um intervalo de 625 μs . Lembrando que em uma rede piconet suporta a interligação de até oito dispositivos sendo que um é o mestre e os outros são os escravos. Piconets se comunicam através de um *gateway*, *bridge* ou um dispositivo mestre comum a ambos os dispositivos. Ao fazer isso, formam o que é chamado de *scatternet* já que é formada por piconets independentes, não sincronizadas que sobrepõe uma mesma área. Uma vez interconectadas estas picotes, que se localizam dentro de uma *scatternet*, formam uma infra-estrutura que torna possível a comunicação de dispositivos que não estão diretamente conectados ou que estão fora de alcance de outro dispositivo. Essa infra-estrutura é chamada de *Mobile Area Network* (MANET). A figura 2.4 ilustra uma *scatternet*.

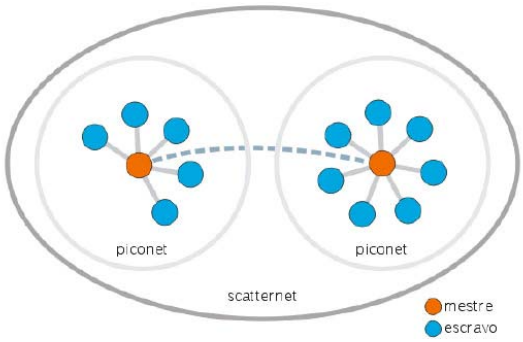


Figura 2.4 – Scatternet

(FONTE: UNICAMP, 2010)

A comunicação entre dispositivos Bluetooth é estabelecida através de um canal FH-CDMA (Frequency Hopping – Code Division Multiple Access) onde os sinais propagados sobre uma faixa de frequência ocupam somente uma pequena largura de banda, ajudando evitar interferências. Foram definidos 79 canais na tecnologia Bluetooth onde a taxa de transferência nominal de cada canal é de 1 MHz. Esses canais também são conhecidos como *hops*, como o nome em inglês do canal acima já insinuou. Uma vez que um dispositivo Bluetooth se conecta com outro dispositivo, a transmissão de dados é realizada por *slots* (pedaços) de tempos. Cada *slot* possui 625 microssegundos de comprimento. Através do Time Division Duplex (TDD), permite a receber e transmitir informações formando assim uma comunicação conhecida por full-duplex. Em uma comunicação full-duplex, os *slots* são utilizados de maneira alternada para o envio e recebimento de pacotes. A figura 2.5 ilustra utilização do canal FH-TDD.

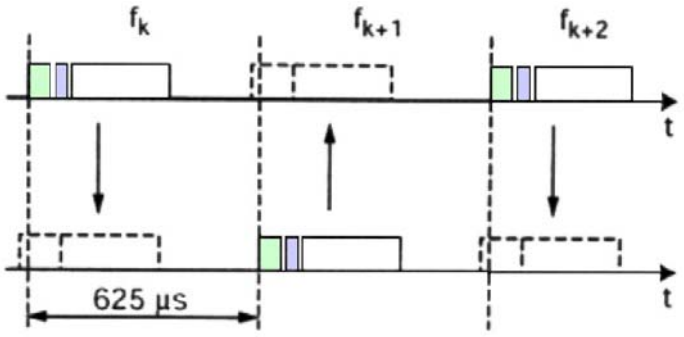


Figura 2.5 – Canal FH/TDD - Bluetooth

(FONTE: SBRC, 2001. Página 778)

Sendo assim, o mestre em uma piconet transmite em slots de tempo pares enquanto que os escravos transmitem em slots de tempo ímpares. Devido ao mecanismo *frequency hopping*, cada slot de tempo, utiliza um canal de frequência diferente. Isso significa que o canal é trocado a cada envio ou recebimento de pacotes. Com isso, apesar de estar em um mesmo recinto, um piconets de um dispositivo Bluetooth não sofrem interferência de outro piconet, pois não conseguem "escutar" a comunicação de outra piconet. (SIG – 2010)

Segundo (MILLER - 2001), há três modos de segurança para um dispositivo Bluetooth. O primeiro é o modo sem segurança (*Non-Secure*), o segundo é o modo de segurança estabelecido no nível do serviço (*Service-Level Enfored Security*) e o terceiro que é o modo de segurança estabelecida no nível do link (*Link-Level Security*). O modo sem segurança é onde nenhuma medida de segurança foi implementada tornando o dispositivo efetivamente inseguro. Já no modo de segurança estabelecido no nível do serviço o dispositivo Bluetooth dá inicio aos procedimentos de segurança depois de estabelecer a conexão. Diferentemente do modo anterior, o modo de segurança estabelecida no nível do link dá inicia os procedimentos de segurança antes mesmo de estabelecer uma conexão. Por esta característica, esse modo é considerado o modo mais seguro. Os procedimentos de segurança que fazem parte da especificação Bluetooth mencionado acima se referem aos mecanismos de *Key Management, Device Authentication* e *Packet Encryption.*

No mecanismo *Key Management,* ou gerenciamento de chave, utiliza-se três tipos de chaves; a chave com código PIN, a chave particular (*Private Link Key*) e a chave de criptografia particular (P*rivate Encryption Key*). A figura 2.6 ilustra a conexão de dois dispositivos. De uma forma resumida, o *Key Management* funciona da seguinte maneira:

- O usuário digita uma senha numérica.
- O dispositivo então gera uma chave de link particular e autentica com o outro dispositivo.
- O dispositivo deriva uma chave de criptografia particular a partir da chave de link e depois autentica com o segundo dispositivo.

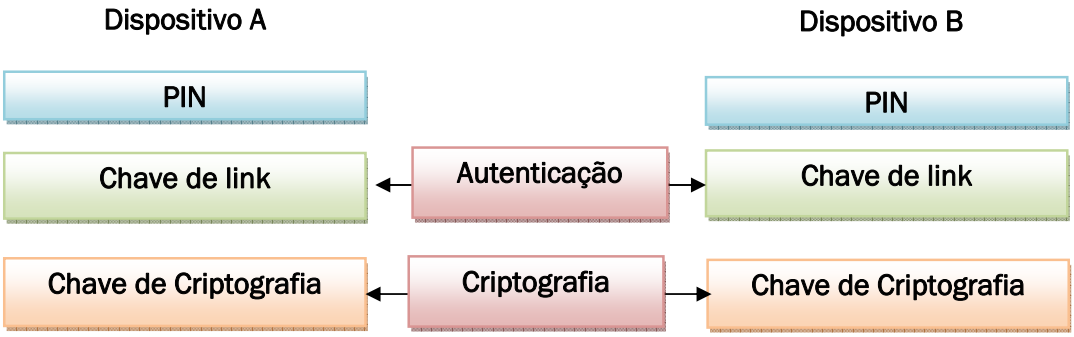


Figura 2.6 – Conexão entre de dois dispositivos

(FONTE: MILLER – 2001, Página 161)

O código PIN (*Personal Identification Number*). Esse número de identificação é escolhido pelo usuário e costuma ser um número de quatro dígitos, que equivalem a 48 bits. A chave particular faz a utilização de um dos quatro tipos de chaves de link, que também são chamadas de autenticação. Essas chaves de links são números aleatórios de 128 bits temporários ou semi-permanentes, gerados a cada transmissão. Uma chave é a de unidade, derivada por um único dispositivo Bluetooth, a outra é a chave de combinação que é derivada de um par de dispositivos. Outra é a chave mestra que é utilizada na situação em que um dispositivo mestre em uma piconet queira transmitir para vários dispositivos ao mesmo tempo. Com isso, substituí-se a chave de link atual por uma sessão. Por fim, a chave de inicialização é aquela usada no processo de inicialização do dispositivo, protegendo então os paramentos de inicialização ao serem transmitidos. (MILLER – 2001, Página 160)

Já para a chave de criptografia particular, a chave deriva da chave de link que está sendo utilizada no momento. Ou seja, ela é sempre alterada. A chave de criptografia varia de 8 a 128 bits de extensão. Essa variação é devido às normas legais com as restrições de exportação de tecnologia de vários países. Além disso, é importante saber que cada dispositivo Bluetooth individual possui seu próprio endereço de dispositivo exclusivo de 48 bits, atribuído pelo IEEE. Contudo, a utilização dessas chaves junto com o endereço do dispositivo gera outras chaves com característica secreta para cada link realizado na conexão. Esse mecanismo garante então que qualquer dispositivo dentro ou fora da piconet, ou scatternet, do dispositivo em uso monitorem uma conexão Bluetooth. (MILLER – 2001, Página 160-161)

O segundo procedimento de segurança que faz parte da especificação Bluetooth se refere ao mecanismo de autenticação de dispositivo*, Device Authentication.* A autentificação de dispositivos consiste no esquema denominado de "desafio e resposta". Este esquema implica que durante a verificação de autenticação entre dois dispositivos se a chave for reconhecida por ambas as parte, a autenticação é realizada. Caso contrário, se alguma das partes não reconhecer a chave, a conexão será cancelada. Esse processo ocorre quando um protocolo de segurança especial é utilizado para verificar se outro dispositivo reconhece a chave simétrica.

E por fim, o procedimento de segurança de criptografia de Pacote, *Packet Encryption*, exige uma criptografia sistemática para cada um dos pacotes que vão ser transmitidos. Esse procedimento é definido por três modos. O primeiro modo de criptografia é similar ao modo sem segurança onde nenhum pacote é criptografado. Já o segundo propõe que o tráfego de dados ponto-a-ponto seja criptografado enquanto o tráfego ponto-a-multiponto não seja. O terceiro e último modo de criptografia exige que todo o tráfego de dados seja criptografado, permitindo então maior segurança na transmissão. (MILLER – 2001, Página 163)

2.1.4 – As funcionalidades e aplicabilidades do Bluetooth

Segundo VINÍCIUS (2008) os três modelos de fones de ouvido com a tecnologia Bluetooth mais concorrida são das marcas Samsung, Motorola e Nokia, de acordo com a figura 2.7. Os três têm conexão garantida de até 10 metros do aparelho que está conectado a ele, devido ao padrão da tecnologia sem fio utilizada. Possuem em torno de 30 gramas cada e uso diário de 6 horas. Foi constatado que o aparelho da Samsung oferece a possibilidade de controle por voz, e por isso se destaca por ser mais intuitivo.

Figura 2.7 – Fone de ouvido da Samsung, da Motorola e da Nokia.

(FONTE: VINÍCIUS, 2008)

A empresa ChinaVasion (página na web: http://www.chinavasion.com/) lançou no dia 25 de abril de 2011 um espelho retrovisor que possui conectividade Bluetooth. Assim, o motorista pode conversar através do aparelho sem precisar segurar o celular ou adquirir um fone de ouvido. Esse retrovisor custa 101,55 dólares e pode ser observado na figura 2.8. Além disso, ele funciona como GPS, player de música e videogame *touchscreen*.

Figura 2.8 – Espelho retrovisor que possui conectividade Bluetooth.

(FONTE: OLHAR DIGITAL, 2011)

De acordo com IKEDA (2011), foi apresentada na Cebit 2011, a Magic Cube. Esse pequeno dispositivo tem o objetivo de projetar um teclado virtual em qualquer superfície plana ou lisa, conforme a figura 2.9. Trata-se de um aparelho que se conecta em celulares ou em *tablets* via Bluetooth facilitando a digitalização de mensagens, devido a projeção de teclas virtuais. Seu preço de custo é de 169 dólares.

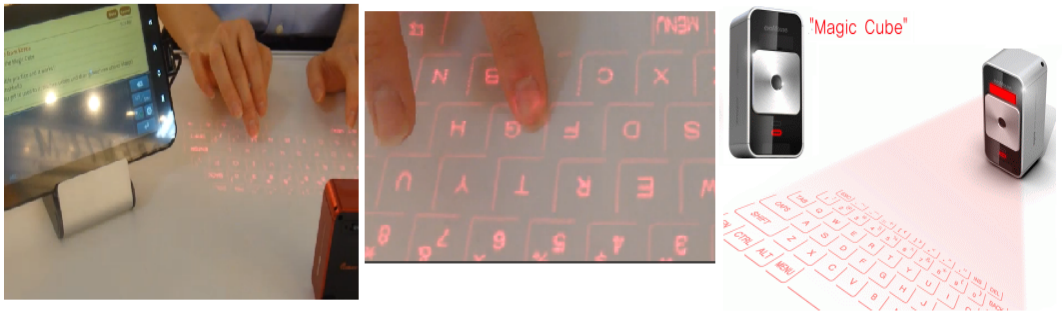


Figura 2.9 – Teclado virtual Bluetooth.

(FONTE: IKEDA, 2011)

A empresa HP (página na web: http://www.hp.com) oferece impressoras com serviços de tecnologia Bluetooth incluídos. O site fornece documentos com instruções gerais para auxiliar a impressão a partir de um dispositivo compatível com Bluetooth. A tecnologia sem fio Bluetooth permite que impressoras, telefones celulares com câmera, PDAs e notebooks se comuniquem a uma distância de até 10 metros. A impressora pode estabelecer uma conexão sem fio Bluetooth com apenas um dispositivo por vez. Os dispositivos Bluetooth podem enviar somente imagens para impressão em impressoras HP com Bluetooth, como na figura 2.10.

Figura 2.10 – Impressora HP com Bluetooth.

(FONTE: HP, 2011)

A empresa The Celluon lançou em 2010 uma nova versão de mouse que utiliza a tecnologia Bluetooth. Esse novo modelo de mouse é chamado de Evo Mouse. Ele funciona em qualquer superfície plana e não requer nenhum objeto físico. Essa nova tecnologia pode ajudar a reduzir lesões adquiridas por movimentos repetitivos causados pelo uso do mouse tradicional.

Figura 2.11 – Evo Mouse, uma nova forma de usar o mouse.

(FONTE: THE CELLUON, 2010)

Seguindo a mesma linha das empresas mencionadas anteriormente, a Motorola lançou um GPS de apenas 4,3 polegadas e que possui as funções básicas de um navegador portátil. Utilizando a tecnologia do Bluetooth, o usuário poderá realizar e receber chamadas no modo viva-voz. No momento que uma ligação é recebida, o aparelho suspende temporariamente as instruções sonoras de navegação para mostrar o número de quem está ligando no visor do GPS.

Figura 2.12 – GPS da Motorola com Bluetooth.

(FONTE: MELLO, 2010)

2.2 – Características do protótipo do portão

O protótipo do portão foi construído em madeira com as dimensões, de 45,2 cm comprimento, altura de 30 cm e 13,5 cm de largura para demonstrar o funcionamento do projeto. A figura 2.13 ilustra o protótipo.

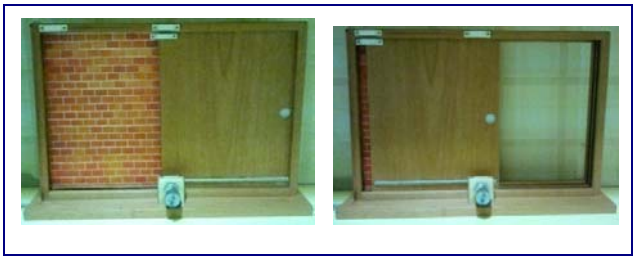


Figura 2.13 – Protótipo de porta deslizante construída para o projeto

Elaborado em madeira, o portão de cor marrom clara possui medidas reduzidas, porém com proporções harmônicas. Foram colocados adesivos de um muro e de um portão para simular a ideia de um portão real. O portão possui uma plataforma retangular também em

madeira, que proporciona a sustentação da estrutura principal. Cortes rentes a madeira foram feitos para a passagem dos fios do LED, sensores de fim de curso e do motor entre a base e a moldura desse protótipo.

O protótipo de portão possui duas partes; uma fixa, para simular a parede, e uma deslizante que se movimenta sobre um trilho de plástico, para simular a abertura do portão. O trilho utilizado para a movimentação do motor foi tirado de duas gavetas de leitora de DVD-ROM da CPU de um computador, como ilustrado na figura 2.14.

Em frente a parte móvel do portão, há um apoio de madeira mais clara onde encontra-se o motor DC reduzido. Este apoio permite o acoplamento e a movimentação do motor em caso de algum problema na parte elétrica, já que é de fácil encaixe. Junto ao motor, há uma rondana localizada na parte central do motor que permite a movimentação do portão.

Figura 2.14 – Trilho da gaveta da leitora de CD utilizado na janela deslizante**.**

2.3 – Telefonia Celular

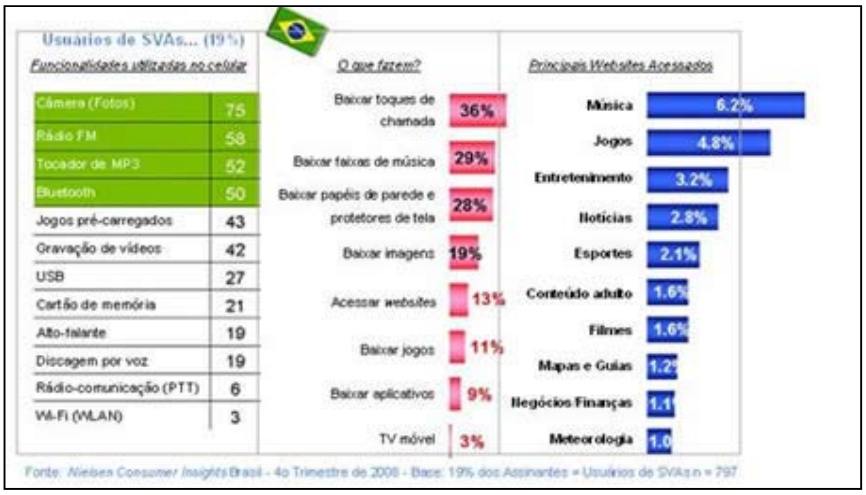
Para criar uma relação entre trabalho proposto e o uso da telefonia celular foi necessário uma pesquisa de mercado.

Conforme divulgação feita em setembro de 2009 pela NIELSEN (2011), o Brasil já é um país do telefone celular. O levantamento feito pela empresa de informações de mercado e consumidor mostra que os brasileiros dividem com o México a liderança no ranking de celulares vendidos na America Latina.

Segundo Thiago Moreira, gerente regional de produtos da Nielsen Telecom Practice Group para a América Latina, a venda de aparelhos com funcionalidades e novas tecnologias aumentam em um ritmo constante na região. Aplicativos relacionados à música, imagem e a novas tecnologias, tais como 3G e Smartphones, também aumentou significativamente e são as

favoritas dos consumidores. O executivo também afirma que os serviços de valor agregado (SVA) também estão na mira desses consumidores.

Uma análise de segmentação de usuários de telefones celulares realizada no Brasil, no último trimestre de 2008, identificou que o uso dos celulares vai além do alô. A pesquisa mostrou que 19% desses usuários utilizam de algum tipo de Serviço de Valor Agregado. Desses 19%, 50% utilizam a funcionalidade Bluetooth, conforme o quadro 2.2. Similarmente, esse estudo também demonstrou que o gasto médio de usuários de serviços de valor agregado é 20% superior ao gasto médio dos usuários apenas de comunicações de voz, segundo Moreira. Com essa pesquisa, pode-se notar que os brasileiros têm preferência por celulares com câmeras, rádio, tocadores de MP3 e Bluetooth.



Quadro 2.2 – Resultados da pesquisa – A preferência dos Brasileiros.

(Fonte: NIELSON - 2011)

Conforme HAMBLEN (2010), além de estar presente em bilhões de celulares, a tecnologia Bluetooth começará a aparecer mais freqüentemente em automóveis e dispositivos utilizados em ambientes industriais e até mesmo médicos. Em 2009, cerca de 984 milhões de dispositivos Bluetooths foram vendidos. A estimativa para 2010 já é de 1,2 bilhões de *gadgets* com a tecnologia sejam vendidos graças ao novo padrão Bluetooth 4.0, que possui baixo consumo de energia.

2.4 – Chave fim de curso

Chaves de fim de curso, ou sensores fim de curso, são auxiliares para tanto para comando quanto para acionamento de dispositivos. Também são conhecidas como *switches.* Um tipo de *switch* é o interruptor de lâminas conhecido como *reed-switch*. O *reed-switch* é composto por duas lâminas pequenas de ferro próximas, contidas dentro de um pequeno invólucro de vidro. Quando um ímã se aproxima, as duas lâminas se aproximam os dois contatos, tornam-se atraídos um pelo outro resultando em contato físico, permitindo a passagem de corrente. Já quando o ímã se afasta, os contatos se desmagnetizam e ao se separar interrompem o circuito. Tal dispositivo é geralmente usado em chaves de fim de curso e alarmes. (ROSÁRIO, 2005, p. 61).

Esses sensores, muito utilizados para detectar o fim de curso em sistemas automatizados para detectar o fim do movimento de um objeto. Seu funcionamento pode ser descrito como liga/desliga de um circuito tendo as lâminas abertas ou fechadas. Na figura 2.12 mostra um desenho conceitual de uma chave de fim de curso. (AQUAHUB, 2010).

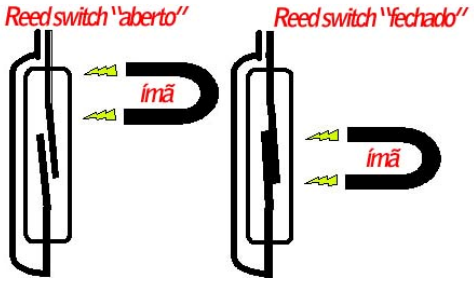


Figura 2.15 – Conceito de uma chave de fim de curso.
(FONTE: AQUAHUB, 2010)

Assim os sensores fim de curso podem ser utilizados para determinar a posição de objetos. Essas chaves sinalizam o início ou fim de um determinado movimento ou posição, liberando fluxo de energia quando acionadas.

2.5 – Arduino

Arduino é um uma plataforma para o desenvolvimento de protótipos de eletrônicos de código aberto, *open-source*. Utilizando uma plataforma baseada em um hardware e software flexível e fácil de usar, o Arduino mostra-se ideal para qualquer pessoa, interessada em criar objetos ou ambientes interativos.

As placas Arduino podem ser tanto construídas a mão quanto comprada pré-montadas. O software da placa e para a programação podem ser baixados gratuitamente do site oficial. Já as informações dos hardwares, como designs de referência (arquivos CAD), são todos disponíveis sob uma licença open-source. A flexibilidade da licença open-source permite adaptá-los às necessidades de seus usuários.

Seu poder de percepção de um ambiente através da entrada de uma variedade de sensores possibilita o controle de luzes, motores e outros atuadores nos arredores deste ambiente. O microcontrolador da placa é programável devido ao uso da linguagem de programação Arduino baseada em *Wiring*. O ambiente de desenvolvimento do Arduino (baseada em *Processing*) permite que projetos em Arduino possam se comunicar com o software executado em um computador ou sozinho, *stand-alone*. Processing é um ambiente de linguagem de programação onde se podem criar imagens, animação até interação. (ARDUINO, 2011)

2.5.1 – Shields

Shields são placas a serem montadas em cima de placa Arduino. Uma vez montadas, elas ampliam a funcionalidade do Arduino para controlar diferentes dispositivos, a aquisição de dados, etc. Essa ampliação ocorre devido a transposição dos pinos da placa Arduino para cima, permitindo então a conexão a uma protoboard, por exemplo.

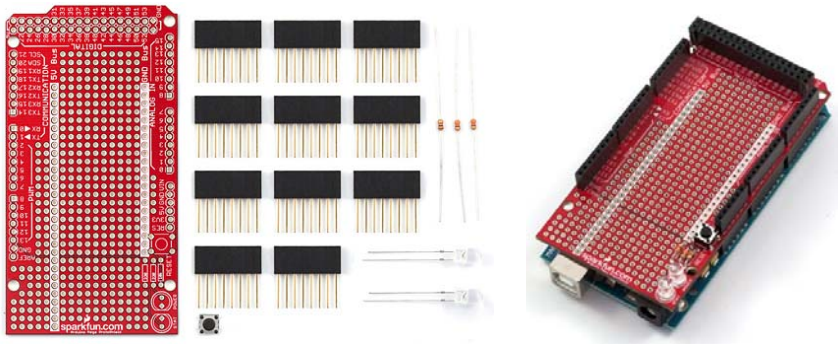


Figura 2.16 – Arduino MegaShield desmontada e montada.
(FONTE: SPARKFUN, 2011)

2.5.2 – Arduino - IDE

A programação para as placas Arduino é feita no IDE desenvolvido únicamente para ele. IDE é a abreviação de *Integrated Development Environment*. Em português significa ambiente de desenvolvimento integrado. O ambiente de programação Arduino é de código aberto o que o torna fácil para desenvolver código e enviá-lo para a placa de E/S.

O IDE Arduino contém um editor que pode ser usado para escrever *sketches*, em português, esboços. *Sketches* é o nome dado aos programas desenvolvidos no Arduino. Esses programas são desenvolvidos em uma linguagem de programação bem simples baseada na linguagem *Processing*.

Figura 2.17 – Logo do programa Arduino Alpha®

Utilizando a IDE, o programa desenvolvido é convertido para linguagem C e depois compilado com AVR-GCC. Esse processo então produz o código binário necessário que o

microcontrolador do Arduino será capaz de compreender e executar seus comandos. Uma vez conectada a um computador, a placa Arduino se comunica através do cabo USB e utilizando a IDE é possível compilar e realizar o *upload* do programa para a placa.

No Arduino, o software denominado Arduino 0022 (versão), pode ser executado em Windows, Mac OS X e até mesmo Linux. Esse ambiente foi desenvolvido em Java e é baseado em *Processing*, avr-gcc, entre outros softwares de código aberto. Essa IDE, também contém diversos exemplos prontos que facilitam o desenvolvimento do código. (ARDUINO, SOFTWARE. 2011)

A figura 2.18 ilustra o software com a compilação de um de seus exemplos já prontos.

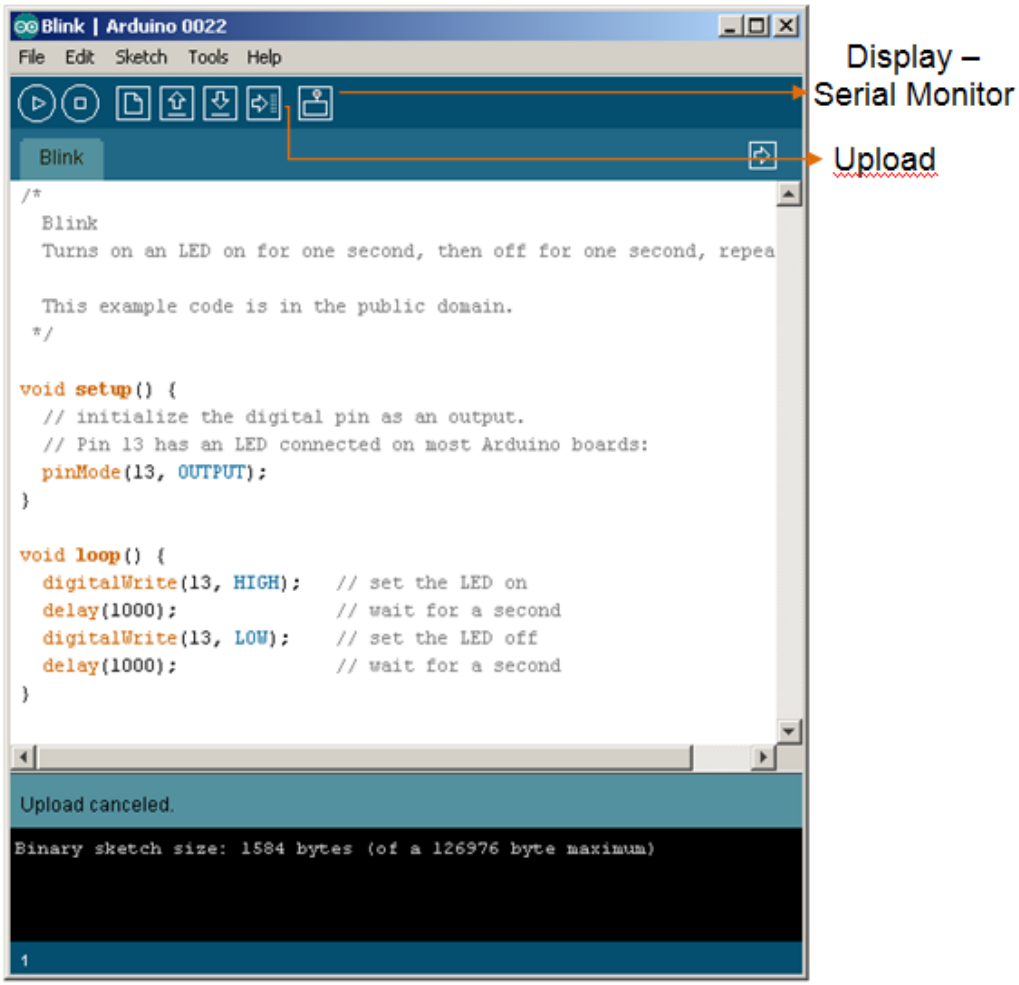


Figura 2.18 – IDE Arduino.

2.6 – Motor DC de corrente contínua (CC)

A ideia de um motor elétrico é transformar energia elétrica em energia mecânica. Dentro os diversos tipos de motores elétricos, o motor DC tem como principal característica o controle preciso da velocidade. (FRANCISCO. 2009, página. 3)

Motores elétricos são encontrados nas mais variadas formas e tamanhos, cada qual apropriado á sua funcionalidade. Motores de corrente contínua possuem diversas aplicações nos setores industriais e até mesmo nos setor residêncial. CONFORME FITZGERALD, KINGSLEY & UMANS (2008, p. 343), as máquinas CC "caracterizam-se por sua versatilidade e pela relativa simplicidade dos seus sistemas de acionamento." Com isso em mente, TORO (1994, página. 325) afirma que "o motor DC oferece uma vasta gama de controle de velocidade e torque, assim como excelente aceleração e desaceleração." Sendo assim, é freqüentemente utilizado na indústria devido à sua facilidade de controle.

Os motores de corrente contínua devem ser ligados a uma fonte de alimentação para poder funcionar. É devido à polaridade da fonte que o sentido de rotação do eixo do motor é determinado. Quando a polaridade é alterada, o motor girará para o lado contrário do que estava girando anteriormente. Já sua velocidade é determinada pela tensão fornecida pela fonte de alimentação. Essa rotação, por sua vez, tem um movimento suave e contínuo.

Também fazem parte do motor CC uma parte fixa e outra parte rotatória. A parte fixa do motor é denominada de estator e não possui nenhum movimento. Sua principal função é produzir um campo magnético constante, através de eletroímã ou um ímã permanente. A segunda parte é a parte rotatória denominada de rotor. A mesma possui um bobinado no qual a corrente elétrica contínua circula.

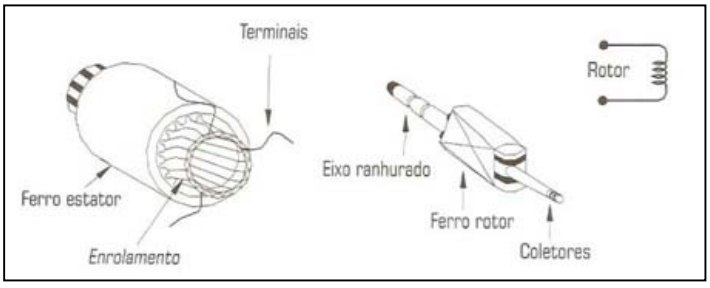


Figura 2.19 – Estrutura interna de um motor DC de ímã permanente.

(FONTE: PAZOS, 2002, p. 169)

Produzindo um campo eletromagnético constante, o estator permite que uma corrente circule através da bobina do rotor, que produzirá uma força que resultará no movimento dos condutores da bobina do rotor. Por sua vez, o comutador troca o eletroímã, que é o campo de posição, de tal maneira que aciona o motor. (PAZOS, 2002, página. 185 e 191).

Na figura 2.20 ilustra-se o princípio de funcionamento da rotação dos motores DC de corrente contínua.

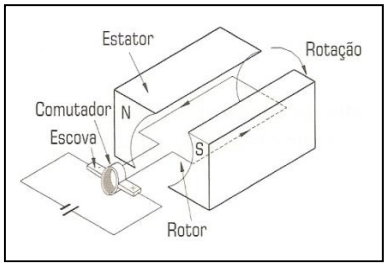


Figura 2.20 – Princípio de funcionamento do motor de corrente contínua**.**

(FONTE: PAZOS, 2002, página. 192)

2.6.1 – Motor DC Reduzido

Como mencionando anteriormente, a velocidade de um motor é determinada pela tensão fornecida pela fonte de alimentação. O objetivo de um controlador de velocidade do motor é dar um sinal que representa a velocidade exigida, acionando então o motor a velocidade solicitada. Existem vários tipos de motores e a saída do controlador de velocidade dependerá dessas variedades. A grande vantagem de um controlador de velocidade segundo HILLS (2005), é que são facilmente encontrados em brinquedos e tem um custo baixo e podem até serem adquiridos em qualquer sucata ou lojas de eletrônicos. Estes motores são geralmente enrolados em série, o que significa para revertê-las.

Figura 2.21 – Motor reduzido12V Kinmore

(FONTE: KINMORE, 2011)

A figura 2.22 ilustra um diagrama de blocos simples de um controlador de velocidade.

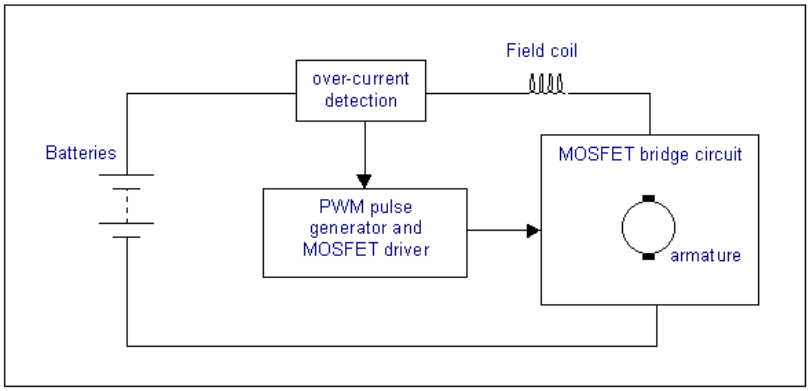


Figura 2.22 – Diagrama de blocos simples de um controlador de velocidade**.**

(FONTE: HILLS, 2005)

2.6.2 – Ponte H

Com intuito de controlar eletronicamente o sentido do giro de um motor DC, diversas soluções podem ser implementadas. Uma das soluções mais conhecida e utilizada é a Ponte H. A origem do nome "Ponte H" é derivado da topologia do circuito, que quando montado, é lembra a letra H, como pode ser observado na figura 2.23. Este circuito pode ser adquirido comércio ou pode ser construído por meio componentes.

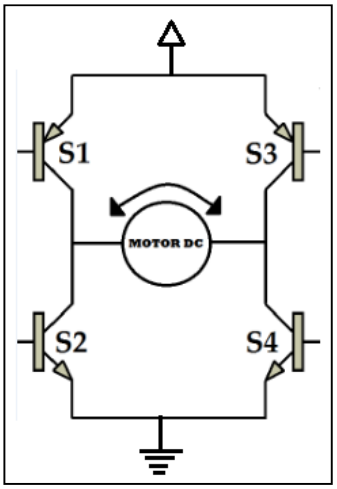


Figura 2.23 – Exemplificação do circuito eletrônico de Ponte H.

(FONTE: Brites & Santos 2008)

Com o intuito de gerar a potência necessária para o motor funcionar, a solução da Ponte H foi utilizada neste projeto. Devido a sua diversidade e ao curto tempo para a elaboração do projeto, o circuito da Ponte H foi adquirido em uma loja de eletrônicos no formato de um circuito integrado - CI. O CI utilizado foi o L293D. O CI L293D é uma ponte H dupla, isto é, controla até dois motores e faz com que estes rodem nas duas direções.

A figura 2.4 ilustra o CI L239D.

Figura 2.24 – Circuito Integrado L293D

De acordo com BRITES & SANTOS (2008, p. 8), a Ponte H, é um circuito eletrônico que permite trocar o sentido de rotação dos motores para ambos os lados. O circuito de Ponte H possui quatro "chaves" rotuladas de S1, S2, S3 e S4. Essas chaves podem ser acionadas alternadamente conforme o acionamento da combinação S1 e S4 ou S2 e S3 conforme a exemplificação do quadro 2.3.

S1	S2	S3	S4	Resultado
1	0	0	1	Motor move para a direita (sentido anti-horário).
0	1	1	0	Motor move para a esquerda (sentido horário).
0	0	0	0	Motor sem energia.
1	1	0	0	Situação não permitida (curto circuito).
0	0	1	1	Situação não permitida (curto circuito).

Quadro 2.3 – Posicionamento das chaves da Ponte H e seus resultados.

Esse CI é necessário, pois motores DC geralmente consomem uma corrente maior do que a suportada pela Arduino (40mA), o que poderia queimá-la instantaneamente. Como a placa Arduino Mega não consegue fornecer a corrente necessária para fazer o motor girar. Com isso, a Ponte H a voltagem indispensável para o acionamento do motor. Similarmente, a ponte H também permite a polarização do motor, o que resulta em um giro tanto para o sentido horário como para o anti-horário. Com a configuração das chaves S1 e S4 fechadas o motor irá

girar no sentido anti-horário. E ao desligá-las, as chaves S2 e S3 são ligadas fazendo com que o motor gire no sentido inverso, no sentido horário. Conforme o quadro 2.1 e a figura 2.23 acima, as chaves S1 e S2, assim como as chaves S3 e S4, são chaves que se encontram do mesmo lado da ponte. Portanto não podem ser ligadas ao mesmo tempo, o que resultaria em um curto circuito do sistema.

CAPÍTULO 3 – DESCRIÇÃO DE HARDWARE

Neste capítulo é detalhada a especificação da placa Arduino Mega, configuração da mesma, do módulo Bluetooth Mate assim como o do motor DC. Também explica-se os circuitos necessários para a montagem do protótipo desse projeto. O hardware utilizado para a construção do presente projeto é composto por: fonte de alimentação, Arduino Mega, Arduino MegaShield, dois breadboards, módulo Bluetooth Mate e um motor DC.

Conforme mencionado anteriormente no capítulo 1, cada componente possui uma função fundamental para o projeto, conforme a descrição abaixo.

- **Arduino Mega:** microcontroladora baseada no processador AVR da Amtel; modelo atmega168 (16kb) controla todo sistema;
- **Motor DC**: motor de corrente contínua (Direct Current). Esses motores giram para ambos os lados, bastando inverter a polaridade da corrente para mudar o sentido da rotação. Permitindo então abrir e fechar a porta deslizante;
- **L293D - Ponte H**: Circuito Integrado (CI) utilizado para controlar motor de corrente contínua. Cada CI deste tipo pode controlar até dois motores DC. Este tipo de controle é necessário, pois motores DC geralmente consomem uma corrente maior do que a suportada pela Arduino (40mA), o que poderia queimá-la instantaneamente.
- **Módulo Bluetooth** (BlueSMiRF Gold): módulo para comunicação sem-fio com o smartphone via Bluetooth.
- **Protoshield - MegaShield**: placa modular que encaixa na Arduino; usada para o desenvolvimento de protótipos (portanto, eliminando a necessidade de fazer soldas);
- **Protoboard**: é uma placa para unir fios sem solda;

3.1 – Placa Arduino Mega

A tecnologia Arduino possui diversos tipos de placas e cada uma delas possui características específicas para diferentes necessidades. Por sua vez, a placa Arduino Mega é

uma versão mais aprimorada da placa Arduino Duemilanove. Graças a sua conectividade via USB, a placa pode ser conectada a qualquer computador com uma entrada USB padrão.

A Arduino Mega utilizada neste projeto é uma placa do microcontrolador que tem como base no ATMega1280. Esta placa possui 54 pinos de entrada digital ou de saída sendo que 14 desses pinos podem ser utilizados como saídas PWM. Também possuem 16 entradas analógicas, 4 UARTs (portas seriais de hardware), um oscilador de cristal 16 MHz, uma conexão USB, uma tomada para fonte de alimentação, um conector ICSP, além de um botão de reset.

Em outras palavras, a placa já vem equipada com todos os dispositivos eletrônicos para suportar o microcontrolador. Para programá-lo, basta conectá-lo a um computador com um cabo USB ou até mesmo ligá-lo a um adaptador AC-DC ou bateria para ligá-lo. A Arduino Mega é compatível com a maioria dos *shields* projetado para o Arduino Duemilanove ou Diecimila. A figura 3.1 ilustra alguns dos componentes da placa Arduino ATMega utilizada neste projeto. (ARDUINO, HARDWARE. 2011)

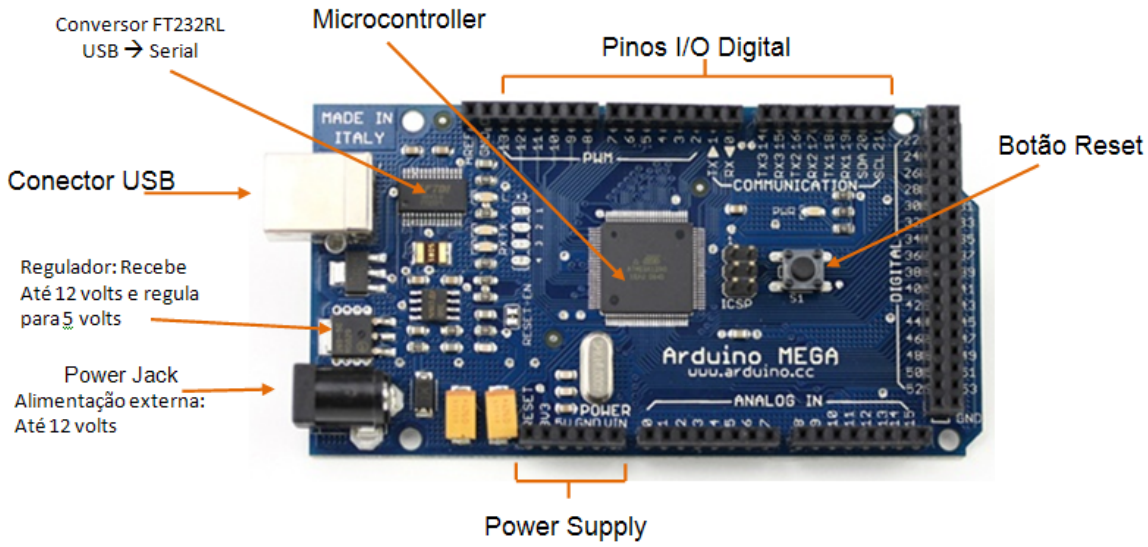


Figura 3.1 – Placa Arduino ATMega1280.

3.1.2 – Fonte de alimentação da Arduino Mega

A fonte de energia do Arduino Mega é selecionada automaticamente. A fonte de alimentação pode ser externa ou a placa pode ser alimentada via conexão USB. Já para alimentação externa, a placa pode ser conectada a um adaptador AC-DC de parede ou até mesmo uma bateria. Se utilizado um adaptador, este pode ser conectado através do *plug* de 2,1 milímetros de centro-positivo na tomada da placa de potência. Caso utilizando uma bateria, os pinos GND e Vin da parte POWER devem ser conectados.

A placa pode operar com uma fonte externa de 6 a 20 volts. Para este projeto foi utilizado uma fonte de 9v para satisfazer tanto as necessidades da placa quanto a do motor. É importante observar que o pino de 5v pode vir a fornecer menos de cinco volts. Com isso em mente, se a fonte utilizada fornecer menos de 7v, a placa pode vir a ficar instável. No entanto, se for utilizado mais do que 12v, o regulador de voltagem da placa pode superaquecer e danificar a placa. Portanto a faixa de alimentação recomendada é de 7 a 12 volts.

Os pinos de alimentação são os VIN, 5v, 3v3e o GND. O VIN é a tensão de entrada para a placa Arduino, quando utilizar uma fonte externa de energia ou a conexão UBS (5v). O pino 5V é a fonte de alimentação regulada usada para alimentar o microcontrolador e outros componentes na placa. Essa energia pode ser proveniente tanto do pino VIN, através de um regulador de bordo, quanto pela conexão USB ou até mesmo de outra fonte de 5V regulada. Já o pino 3V3 é a fonte de 3,3 volts gerada pelo chip FTDI on-board, cuja corrente máxima é de 50 mA. E por último o pino GND é o pino terra. (MEGA, 2011)

3.1.3 – Características e Comunicação da Arduino Mega

O Arduino Mega possui uma série de facilidades para se comunicar com um computador, outro Arduino, ou até mesmo outros microcontroladores. O ATMega1280, modelo utilizado no projeto, fornece quatro UARTs hardware para comunicação TTL (5V) de série. A placa também é composta por uma porta COM virtual para o software no computador. Outras características da placa podem ser observadas no quadro 3.1. Para melhor visualização, a IDE, ou software, inclui um monitor Arduino serial que permite que dados simples de texto possam ser enviados para e de placa Arduino. E por fim, os LEDs RX e TX na placa piscam quando dados estão sendo transmitidos pelo chip FTDI e pela conexão USB para o computador, no entanto não piscam para comunicação serial nos pinos 0 e 1. (MEGA, 2011)

Microcontrolador	ATmega1280
Tensão	5V
Tensão de entrada (recomendado)	7-12V
Tensão de entrada (limites)	6-20V
Pinos de E/S Digital	54 (dos quais 14 oferecem saída PWM)
Pinos de entrada analógica	16
Corrente DC por Pino de E/S	40 mA
Corrente DC para o Pino 3.3V	50 mA
Memória Flash	128 KB no qual 4 KB é utilizado pelo bootloader
SRAM	8 KB
EEPROM (memória)	4 KB
Velocidade de relógio	16 MHz

Quadro 3.1 – Características da Arduino Mega.

3.1.4 – Microcontrolador ATMega1280

O Arduino Mega utilizado no projeto foi a ATMega 1280. O microcontrolador AT1280 da ATMEL é um microcontrolador de baixo consumo de energia baseado na arquitetura reforçada AVR RISC. Sua grande vantagem é executar instruções em um ciclo único, otimizando então a velocidade de processamento ao invés do consumo de energia. Este microcontrolador é considerado ser o coração da Arduino Mega.

Device	Flash	EEPROM	RAM	General Purpose I/O pins	16 bits resolution PWM channels	Serial USARTs	ADC Channels
ATmega1280	128KB	4KB	8KB	86	12	4	16

Quadro 3.2 – Configurações do AT1280.

(FONTE: ATMEL, 2011, p. 7)

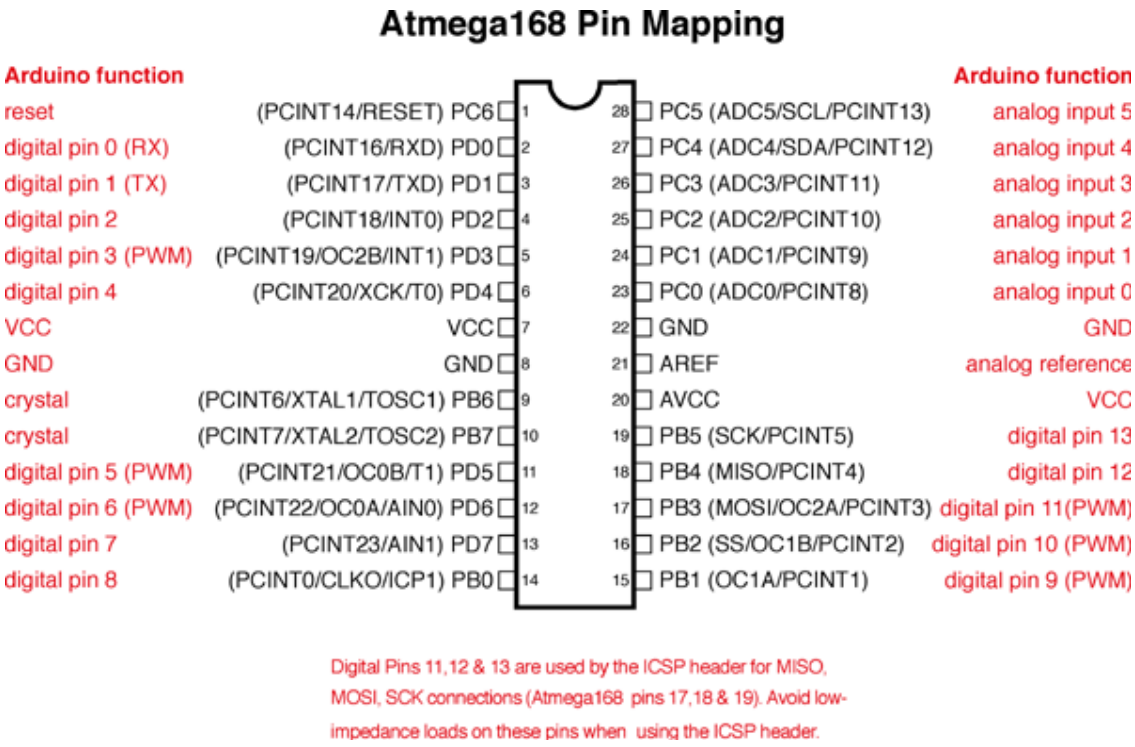


Figura 3.2 – Dentro do MCU do AT1280.

(FONTE: HACK, 2011)

Algumas de suas características são:

• possui arquitetura RISC

• 20 MIPS (20 Milhões de instruções por segundo)

• 16Kb Flash / 512 b EEPROM / 1Kb RAM Estática

• 10.000 ciclos na Flash e 100.000 na EEPROM

• 2 contadores / temporizadores de 8bits

• 1 contador / temporizador de 16bits

• 1 temporizador de tempo real com clock a parte

• 14 portas digitais

• 6 portas analógicas

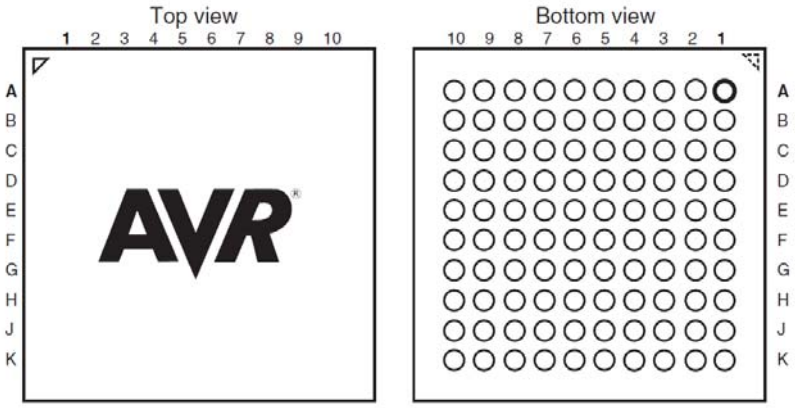


Figura 3.3 – Visão da Pinagem do AT1280.

(FONTE: ATMEL, 2011, p. 3)

3.2 – Especificações e pinagem do CI – L293D

Uma vez entendido o conceito da ponte H, é importante lembrar que o CI L293D possui dois lados, um controla o motor 1 e outro controla a velocidade do motor. A figura 3.4 ilustra o a pinagem deste CI. Já a figura 3.5, ilustra o esquemático simplificado deste CI.

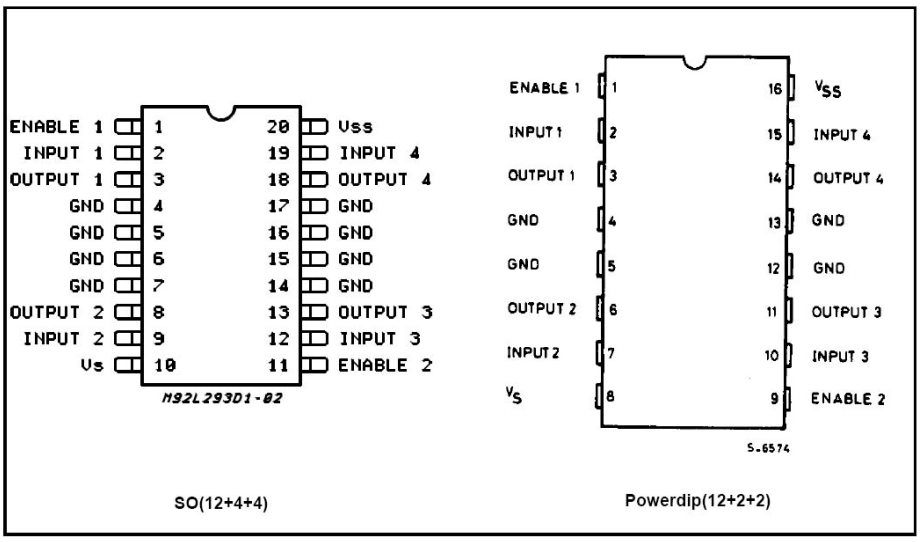


Figura 3.4 – Pinagem L293D.

(FONTE: THOMPSON, SGS. 1996).

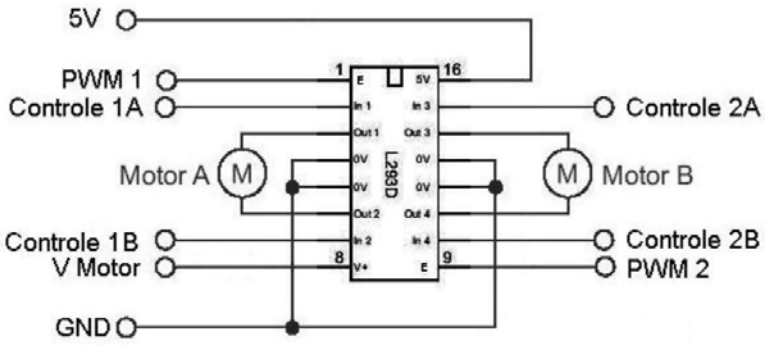


Figura 3.5 – Esquemático simplificado L293D para o controle de 2 motores.

(FONTE: BEBOP, 2011)

Na figura 3.5 pode-se observar:

- **Enable/PWM (motor 1: pino 1; motor 2: pino 9):** liga/desliga o motor. Também pode ser utilizado para regular a velocidade do motor com PWM.
- **Control/Input (motor 1: pinos 2 e 7; motor 2: pinos 9 e 15):** cada motor possui 2 pinos de controle. Eles funcionam de acordo com o quadro 3.1 abaixo.
- **Motor/Output (motor 1: pinos 3 e 6; motor 2: pinos 10 e 16)**: aciona os motores de acordo com o estado dos pinos de controle; O motor deve ter um fio em cada output (são apenas dois por motor).
- **VCC2**: tensão que será utilizada pelos motores. Arduino trabalha com apenas 5v, no entanto motores podem requerer uma tensão de funcionamento maior. No circuito utilizado neste projeto foi utilizada uma fonte de 9v provenientes da placa Arduino Mega. * *Como a fonte de alimentação é externa foi necessário conectar o neutro (GND) da fonte externa com o da Arduino para não queimá-la*.
- **VCC (pino 16)**: tensão para alimentar o CI, 5v. Podendo-se usar os 5v da Arduino.
- **GND (pinos 4, 5, 12 e 13)**: neutro/negativo/terra/gnd. Devem estar conectados no 'GND' da Arduino. **Novamente, como no circuito utilizado neste projeto foi utilizado uma fonte de alimentação externa de 9v, foi necessário conectar o neutro (GND) da fonte externa com o da Arduino para não queimá-la*.

INPUT A1	INPUT A2	DIREÇÃO
HIGH	LOW	Frente
LOW	HIGH	Trás
LOW	LOW	Parada Rápida
HIGH	HIGH	Parada Rápida

Quadro 3.3 – Controle de INPUT do motor e o acionamento com o CI

(FONTE: BEBOP, 2011)

3.3 – Módulo Bluetooth Mate Gold

O módulo Bluetooth Mate Gold é um dispositivo que utiliza a tecnologia sem fio Bluetooth. O Bluetooth Mate foi projetado especificamente para ser usado com Arduino. Esse dispositivo funciona como um modem serial (RX/TX) e é substituto sem fio para cabos seriais. Uma de suas características é que o mesmo suporta um fluxo serial de 9600 até 115200bps, podendo ser transmitido diretamente de um computador para seu destino. No modelo GOLD, pode-se transferir dados em uma distância de até 100 metros. Já o modelo SILVER abrange somente 30 metros. A figura 3.6 ilustra o módulo.

Figura 3.6– Módulo Bluetooth - BlueSmirf Mate Gold

Bluetooth Mate possui o mesmo pino de saída, como o Basic FTDI, que destina-se ao encaixe direto na placa Arduino Mega. Ele também é composto por uma unidade denominada de RN-41, módulo Bluetooth de classe 1. Módulos Bluetooth de classe 1 possuem um menor alcance do que os de classe 2, como o Bluetooth Mate Silver mencionado anteriormente.

O módulo Bluetooth Mate possui reguladores de tensão *on-boards*, ou seja, na própria placa. Esses reguladores podem ser alimentadores por qualquer tipo de fonte externa de 3,3 a 5vDC. Com isso os pinos RX/TX são tolerantes a essa tensão. No entanto este dispositivo não pode ser conectado a uma porta serial. Para isso, seria necessário um conversor R232 para TTL para poder conectá-lo a um computador. A placa Arduino Mega utilizada neste projeto possui tal conversor. A figura 3.7 mostra os descritos.

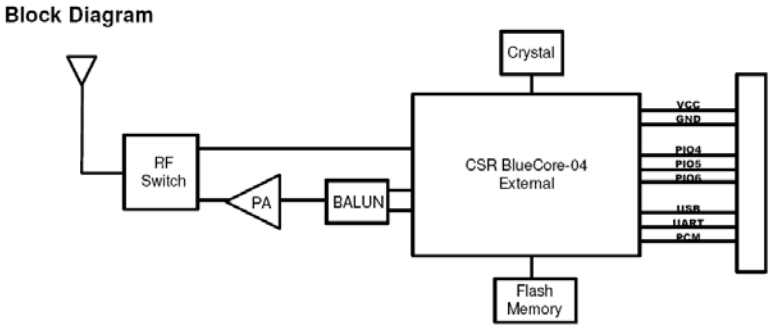


Figura 3.7– Diagrama de blocos do Módulo Bluetooth Mate Gold

Algumas características do módulo Bluetooth incluem:

- Projetado para trabalhar diretamente com Arduino Pro e placas principais LilyPad
- Aprovado FCC Classe 1 Bluetooth ® Modem Rádio
- Link de transmissão muito robusto tanto em sua integridade quanto na distância de (100 metros) - sem derrapagens ou mais buffer.
- Baixo consumo de energia: 25mA avg
- A frequência de Hardy hopping regime - opera em ambientes agressivos RF como Wi-Fi, 802.11g, e Zigbee
- Conexão criptografada
- Frequência: 2,4 ~ 2,524 GHz
- Tensão de funcionamento: 3.3V-6V
- Comunicação serial: 2400-115200bps

- Temperatura de operação: -40 ~ +70 C
- Antena Interna
- Dimensões: 1.75x0. 65 polegadas

O módulo Bluetooth utilizado neste projeto foi responsável pela pesquisa de dispositivos Bluetooth ativados na área. Graças ao seu alcance de 100 metros, todos os dispositivos ativados são encontrados. Possuindo 6 pinos, somente 4 pinos foram utilizados para a ligação conforme a quadro 3.4.

BlueSmirf	Arduino
GND	GND
VCC	5v
RX	TX
TX	RX

Quadro 3.4– Ligação BlueSmirf Gold

3.4 – Chave de Fim de Curso

A chave de fim de curso teve um papel fundamental neste projeto. Seu papel é monitorarar o posicionamento da porta deslizante. Fácil de instalar, 3 chaves foram instaladas estrategicamentes na moldura da porta deslizando com o uso de cola quente.

3.4.1 – Detalhamento da Chave de Fim de Curso

Do tipo *Magnetic Reed-Switch,* as chaves fim de curso utilizadas neste projeto foram adquiridas em pares. Um par possui um lado com um ímã e o outro possui um encapsulamento de plástico branco que protege sua conexão interna normalmente aberta (NA) conforme a ilustração na figura 3.8.

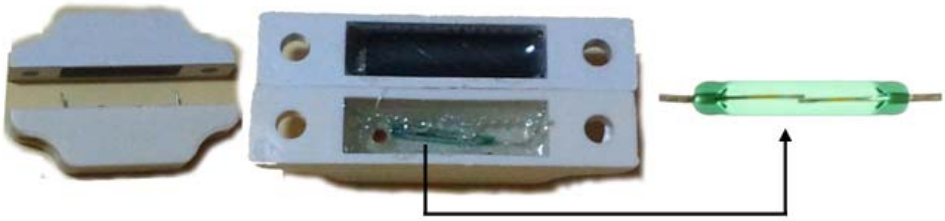


Figura 3.8 – Chave de fim de curso *reed-switch*.

O acionamento do circuito ocorre devido ao posicionamento as palhetas seladas na cápsula. O *reed-switch* utiliza uma chave magnética de palheta. Essa chave consiste em dois contatos selados em uma atmosfera de gás inerte seco dentro de uma cápsula de vidro. Essa cápsula protege o contato das palhetas de contaminação. As palhetas seladas na cápsula são separadas por uma pequena abertura de ar, no entanto suas extremidades se sobrepõem (REED-SWITCH INFO, 2011).

Na figura 3.9, é possível observar as extremidades das palhetas sobrepostas.

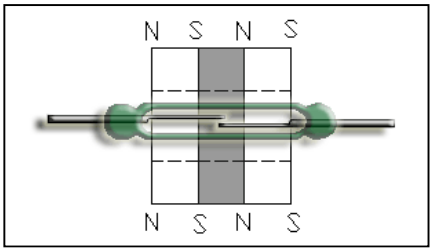


Figura 3.9 – Magnetização da chave de fim de curso.

(FONTE: REED RELAYS AND ELECTRONICS, 2009)

Para poder permitir a passagem de corrente, basta a aproximação de um ímã aos contatos dentro da cápsula de vidro tornando-os atraídos um pelo outro, resultando então no e se tocam, permitindo o toque das palhetas. Mas ao afastar o ímã, as palhetas são desmagnetizadas, causando a separação das mesmas, interrompendo então o circuito (AQUAHUB, 2010).

Devido a caracteristica desse funcionamento, dois sensores fim de curso com a cápsula de vidro e as palhetas foram posicionados no final e no meio da porta deliszante e outro com o ímã foi colocado paralelo ao sensor do meio da porta.

3.5 – Motor DC com Redução

O motivo pelo qual foi utilizado um motor com redução é devido ao fato que a velocidade de saída de um motor normalmente é muito rápida para o uso do cotidiano. Como maioria dos motores de corrente contínua possuem tensões de operação que giram desde 1.000 rpm (rotações por minuto) ou até mesmo o mais do que 50.000 rpm, como os motores *brushless* DC, foi necessário controlar a taxa na qual o motor giraria para evitar que o portão quebrasse. (DAWSON, 2011)

O motor DC com redução, mostrado na figura 3.10, foi utilizado para fazer a força mecânica necessária para abrir e fechar o portão do protótipo. Esse motor foi utilizado por se adequar as características de potência e conexões necessárias para o funcionamento do projeto.

Figura 3.10 – Motor DC 12v com redução utilizado

3.5.1 – Detalhamento do Motor DC com Redução

O modelo do motor utilizado foi o de 12v e 18rpm com o número de registro 25JA3K/2420-1252, conforme a figura 3.11. Tal motor é bastante utilizado usado em brinquedos de crianças e protótipo eletrônicos. Seus dados técnicos são demonstrados no quadro 3.3 abaixo.

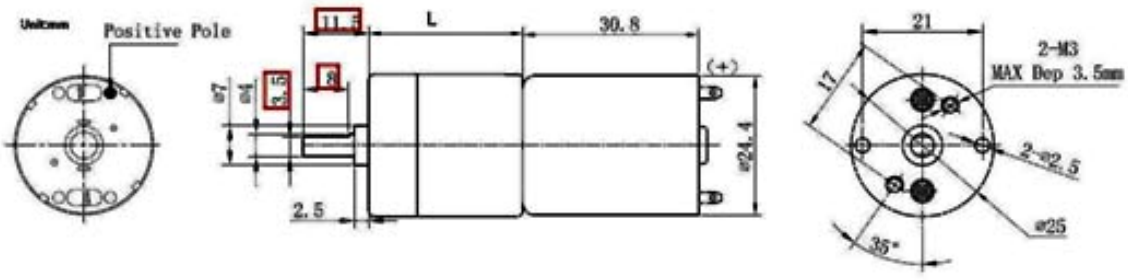


Figura 3.11 – Medidas em milímetros do motor DC.

(FONTE: KINMORE, 2011)

Dados Técnicos:
Tensão de funcionamento: 12V
Nenhuma velocidade da carga: 94rpm
Sem corrente de carga: Max 76mA
Velocidade da carga avaliada: 92rpm
Classificado carga atual: Max 96mA
Torque avaliado: 0.22kg.cm

Quadro 3.5 – Dados técnico do motor DC com redução.

(FONTE: KINMORE, 2011)

3.6 – Comando AT

Os comandos *Hayes AT Commands,* mais conhecidos como comandos AT, são utilizados para controlar, configurar e solicitar serviços nos modems. Em outras palavras, eles são um conjunto de linguagem de comandos ou instruções que são enviadas para um módulo de celular para realizar ações diferentes. Esses comandos ou instruções podem ser enviados ao longo de um serial, USB, ou outra interface para os módulos de celulares a partir do controlador principal da aplicação.

Neste projeto foi utilizado o Arduino IDE instalado em um computador para possibilitar as configurações do programa desenvolvido com os comandos AT assim como seu envio. A comunicação entre a Arduino Mega e o computador foi estabelecida via conexão USB.

3.7 – Endereço MAC

Do inglês *Media Access Control,* o endereço MAC é um endereço único de um dispositivo. Esse endereço é tão único que no mundo não há duas placas com o mesmo endereço. Um endereço MAC é um endereço físico de 48 bits representado por 12 dígitos hexadecimais agrupados dois a dois. Esses grupos são separados por dois pontos. Um exemplo seria 04:18: 0F: F6: 95:6C. Os três primeiros octetos são destinados à identificação do fabricante, os três posteriores são fornecidos pelo fabricante.

Neste projeto o endereço MAC do celular foi utilizado para identificar o usuário. Como seu endereço é único, nenhum outro dispositivo terá o mesmo endereço, portanto o mesmo será como o documento de identidade de seu portador.

3.8 – LEDs

Do inglês, *Light Emitting Diode* os LEDs são excelentes para utilização em protótipo. Neste projeto foi utilizado um LED RGB para indicar os três estados da porta deslizantes. A sigla RGB significa as cores vermelho verde e azul em inglês. Com um LED RGB é possível acionar estas cores e/ou a combinação das mesmas. Este LED possui quatro pinos 5 milímetros, sendo o Cátodo é o maior dos pinos e os outros são designados uma para cada cor.

Como ilustrado na figura 3.12 abaixo, o pino 1 corresponde à cor vermelha, enquanto o pino 3 corresponde à cor verde e pino 4 a cor azul.

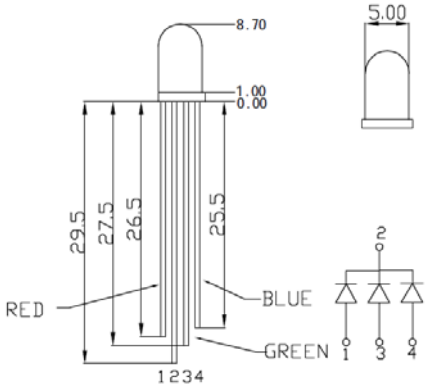


Figura 3.12 – Dimensões LED RGB

(FONTE: LED, 2011)

CAPÍTULO 4 – IMPLEMENTAÇÃO

Neste capítulo são apresentados ao leitor tópicos fundamentais para a compreensão geral da implementação do projeto. Para melhor entendimento da implementação do trabalho, é necessário entender o protótipo do projeto juntamente com seu circuito e seus códigos fonte.

4.1 – Descrição da implementação

Para do início a implementação do projeto, foram definidos conceitos, técnicas e dispositivos utilizados. A implementação consiste em executar as tarefas previstas no planejamento deste projeto.

Uma das primeiras tarefas a ser executada, é estabelecer a fonte de alimentação. Sem energia, não há corrente de energia para a alimentação do circuito o que interfere também no funcionamento do protótipo.

Similarmente a fonte de energia, o cadastramento do endereço MAC do dispositivo Bluetooth na programação do projeto é de grande importância. Para identificá-lo, monitor do Arduino IDE também pode ser utilizado, uma vez que o módulo Bluetooth pesquisa todos os dispositivos próximos, tanto os cadastrados e os não cadastrados, e o demonstram em tela. Uma vez localizado o endereço do dispositivo, o endereço então é adicionado no código do sistema.

O sistema desenvolvido permite que o circuito seja acionado somente com a identificação do dispositivo cadastrado. Uma vez carregado via USB para a placa Arduino Mega, o sistema faz uma pesquisa dos dispositivos visíveis área e aciona o motor caso encontre um cadastrado. Conforme a posição da porta deslizante do protótipo, os sensores fim de curso indicam para o Arduino o estado do motor. A resposta do estado do motor é ilustrada pelo acionamento do LED em diferentes cores assim como também no *display* do monitor no Arduino IDE.

4.2 – Apresentação geral do projeto

Neste projeto, foi desenvolvido um programa em C, que é compilado no Arduino IDE, versão 0022, e carregado via USB para a placa Arduino Mega AT1268, que é responsável pelo

controle de todo o sistema desde a porta deslizante, a leitura dos sensores, acionamento do LED até o acionamento do motor. Na programação cada função foi declarada separadamente.

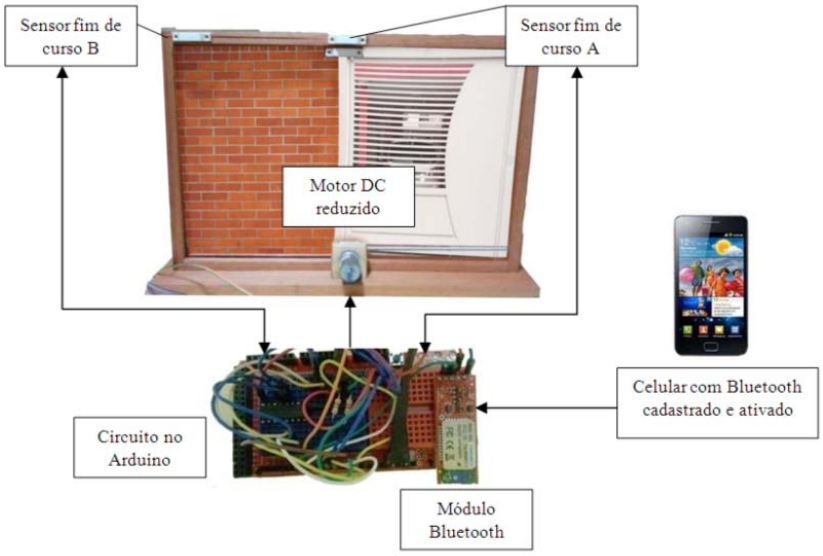


Figura 4.1 – Apresentação do projeto

Como a parte lógica do sistema, o Arduino Mega analisa os resultados encontrados pelo módulo Bluetooth. Após a pesquisa realizada através de comandos AT na programação, o Bluetooth fornecerá uma lista dos endereços MAC dos dispositivos encontrados. Se nenhum dispositivo for encontrado, o monitor mostrará a mensagem “Nenhum dispositivo foi encontrado.”

Com os dispositivos encontrados, a Arduino Mega então verifica se algum desses endereços está na lista de autorizados. Caso tenha encontrado um dispositivo autorizado, a Arduino Mega envia um sinal HIGH para o pino FRENTE_PIN e LOW para o pino TRAS_PIN para o motor acionando-o para abrir a porta deslizante. Caso contrário, o MCU da Arduino voltará a realizar uma nova pesquisa na tentativa de encontrar algum dispositivo autorizado.

Observando figura 4.1, enquanto o sensor A estiver acionado, o LED vermelho permanecerá aceso, indicando que a porta deslizante se encontra fechada. Uma vez acionado, o motor girará para a abertura da porta deslizante. Durante esse período, o LED se tornará azul

até o acionamento do sensor B. Enquanto o sensor não for acionado, isto significa que s porta deslizante ainda está se abrindo.

Uma vez que o sensor B for acionado, o LED se tornará verde. Mas se após 10 segundos do acionamento do motor o sensor B não for acionado, o LED azul ficará piscando indicando que ocorreu algum erro ou travamento da porta. Dessa maneira o usuário saberá que não poderá entrar. Outro controle realizado é que quando aberta, a porta deslizante permanecerá nesse estado por 13 segundos até que o Arduino Mega mande outro comando de LOW para o pino FRENTE_PIN e HIGH para o pino TRAS_PIN para o motor acionando-o para o outro lado para fechar a porta deslizante.

Após a realização de a operação abrir e fechar, o sistema volta a executar uma nova pesquisa de dispositivos. Nessa nova pesquisa é importante ressaltar que se o sistema encontrar um dispositivo autorizado que acionou o circuito a menos de 60 segundos, o sistema informará que não será possível acionar o motor. Esta medida foi tomada para que o motor não seja acionado indevidamente caso o usuário mantenha o Bluetooth ativado a todo tempo. A figura 4.2 ilustra o caminho seguido a partir da pesquisa dos dispositivos.

Figura 4.2 – Diagrama de blocos do projeto

4.3 – Apresentação geral dos circuitos

Neste tópico são abordados todos os circuitos construídos durante a construção do protótipo do projeto.

4.3.1 – Fritzing

Fritzing é um software de código aberto desenvolvido para apoiar os designers, artistas, pesquisadores e criadores a trabalhar criativamente com a eletrônica interativa. Fritzing foi criado em agosto de 2007 no Laboratório de Design de Interação da Universidade de Ciências Aplicadas de Potsdam, na Alemanha onde continua a ser desenvolvido por pesquisadores. Este software tem como inspiração a linguagem de programação *Processing* e Arduino. Esta ferramenta foi desenvolvida com intuito de permitir aos usuários uma maneira prática de documentar seus protótipos e compartilhá-los com os outros. O mesmo também permite criar um layout de circuito impresso para a fabricação. (FRITZING - 2011)

O programa Fritzing® da Fachhochschule Postsdam na versão 0.5.2, conforme ilustrado na figura 4.3, foi utilizado neste trabalho para permitir a configuração e ilustração do circuito do projeto. Sua grande vantagem é possuir uma biblioteca de dispositivos Arduino, o que outros programas de desenho de circuitos não possuem.

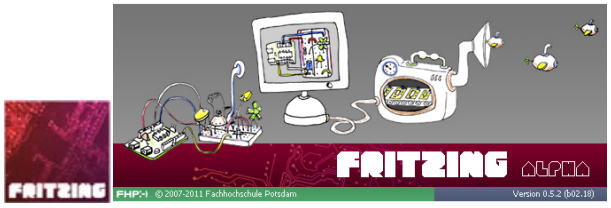


Figura 4.3 – Logo do programa Fritzing Alpha®

4.3.2 – Circuito completo no Fritzing

Para a implementação do circuito do projeto foi utilizado o programa Fritzing®, onde foi possível traçar e simular a melhor disposição dos componentes eletrônicos. As figuras 4.4 e 4.5 ilustram as conexões dos circuitos conectados a Arduino Mega de uma maneira simples e clara.

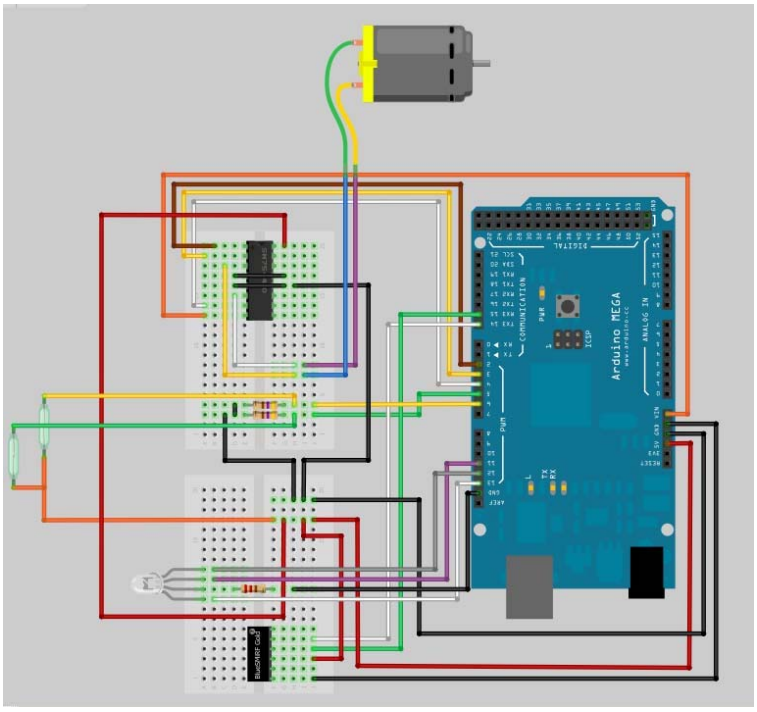


Figura 4.4 – Circuito completo no módulo Breadboard programa Fritzing Alpha®

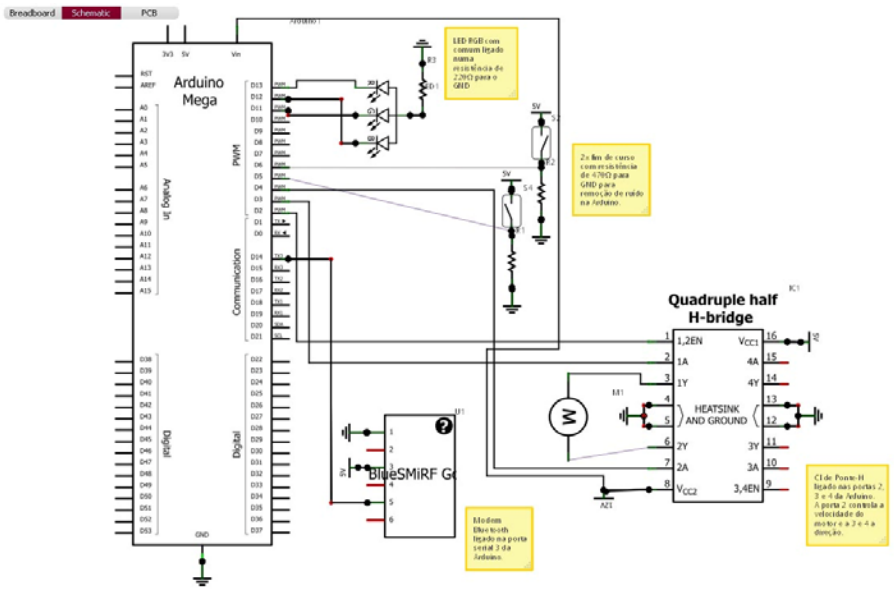


Figura 4.5 – Esquemática do circuito completo programa Fritzing Alpha®

4.4 – Circuito do módulo Bluetooth Mate

A implementação do circuito do módulo Bluetooth BlueSmirf Gold foi simples e sem muitos testes. O datasheet fornecido foi essencial para sua ligação com o resto to circuito. No entanto sua soldagem na placa do protótipo, pois não foi confeccionada uma placa com circuito impresso. Todas as ligações foram feitas através de cabos do estilo *jumper*. A figura 4.6 ilustra o esquemático do circuito do módulo BlueSmirf Gold.

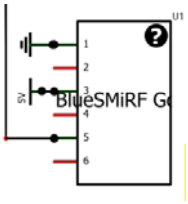


Figura 4.6 – Esquemático do circuito BlueSmirf Gold.

4.5 – Circuito do motor DC

Na implementação do circuito do motor DC o uso do CI L293D foi essencial para o controle do motor. Com a utilização desse CI, não foi necessário construir um circuito de Ponte H, pois este CI possui todos os componentes de um circuito de Ponte H internamente. A figura 4.7 ilustra a conexão do circuito do motor. O CI de ponte H está ligado às portas 2, 3 e 4 do Arduino. A porta 2 é responsável por controlar a velocidade do motor e as portas 3 e 4 pela direção do mesmo.

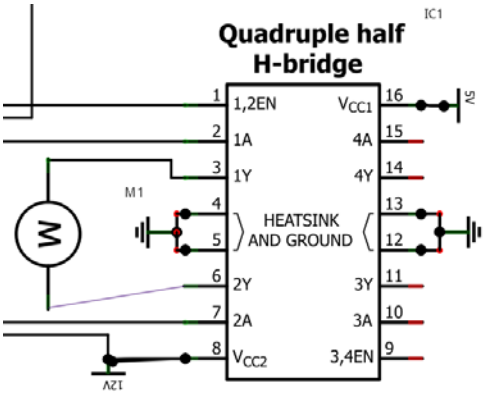


Figura 4.7 – Circuito do motor DC com CI de ponte H

4.6 – Circuito da Chave Fim de Curso com o LED RGB

O circuito da figura 4.8 ilustra as conexões das chaves fim de curso e o LED RGB. O LED RGB tem o pino comum ligado em uma resistência de 220Ω para o GND, terra. Já os pinos indicativos de da cor vermelha foi conectado ao pino PWM 13 do Arduino, o verde ao pino 11 e o azul ao pino 12. Cada pino é acionado conforme a resposta do sensor fim de curso.

As chaves fim de curso possuem uma resistência de 10K Ω ligadas para o GND com o intuito de remover ruídos na placa Arduino. Dessa maneira, o acionamento indevido das mesmas é evitado. Estas foram ligadas nos pinos 5 e 6 do Arduino para comunicação.

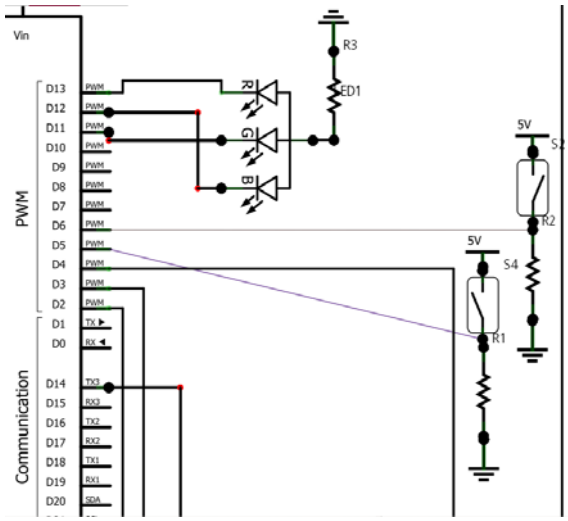


Figura 4.8 – Circuito dos sensores fim de Curso e do LED

4.7 – Detalhamento do código fonte

Para o completo entendimento do funcionamento do sistema desenvolvido, é essencial a compreensão do código desenvolvido para o protótipo de simulação do acionamento do motor.

4.7.1 – Código fonte do protótipo das funções

Como ilustrado na figura 4.9, a pinagem do motor, variáveis de controle de operação e LED são definidos assim como a constante dos dispositivos autorizados a acionar o motor.

```
//Motor esquerdo
const int LIGADO_PIN = 2; //FIO VERDE - ENABLE 1 (PWM)
const int TRAS_PIN = 3; //FIO AMARELO - INPUT 1
const int FRENTE_PIN = 4; //FIO BRANCO - INPUT 2
const int VELOCIDADE_MOTOR = 255;//POTENCIA DO PWM - ENABLE 1

//Variáveis de controle de operação
const int OPENED_SIGNAL = 5;//INPUT - FIM DE CURSO - Fio verde
const int CLOSED_SIGNAL = 6;//INPUT - FIM DE CURSO-  Fio amarelo
const long OPEN_DELAY = 10000;//TEMPO DE ABERTURA DO PORTÃO
const long OPERATION_DELAY = 15000;//TEMPO DE OPERAÇÃO DO PORTÃO

//LEDs
const int LED_VERMELHO = 13; // Fio Branco com a resistencia de 440 ohms
const int LED_VERDE = 11; // Fio Roxo
const int LED_AZUL = 12; //Fio Cinza

/**
 * Retorna todos os dispositivos autorizados da memória separados por ';'.
 * Ex: <mac_address_1>;<mac_address_2>;...
 */
const String AUTHORIZED_DEVICES = "E0F84723E267;0023F1B2ADEA;BC47605475B3;04180FF6956C;2021A5278576;";
```

Figura 4.9 – Constantes e sua pinagem na placa Arduino.

4.7.2 – Código fonte para a pesquisa de dispositivos

De acordo com a figura 4.10, o código permite o envio dos comandos AT ao modem Bluetooth que inicia a pesquisa de dispositivos. O comando INQUIRY = “IN5” pesquisa os dispositivos próximos e retorna somente com o endereço MAC de cada dispositivo encontrado. O INQUIRY_ACK informa que foram encontrados dispositivos próximos e o INQUIRY_NOT_FOUND infoma que a busca foi sem resultados. O delay de 10000 milisegundos é para fornecer tempo suficiente para que a busca seja realisada por completo antes de iniciar uma nova busca.

Outra função importante é a lastAuthorizedDevice. Essa função armazena o último dispositivo a acionar o motor. Ela é importante para garantir que o acionamento do motor não seja indevido.

PortaoEletronico | Arduino 0022

File Edit Sketch Tools Help

PortaoEletronico

```
//Armazena o último dispositivo que abriu o portão
String lastAuthorizedDevice;
long lastOpenTime;
const long LAST_DEVICE_IGNORE_DELAY = 60000;

//Comandos do Bluetooth
String COMMAND_MODE = "$$$"; //Comando para entrar em modo de comando
String DATA_MODE = "---"; //Comando para sair do modo de comando
String CR = "\r\n"; //Quebra de linha (Carriage Return)
String INQUIRY = "IN5"; //Comando para pesquisar os dispositivos em volta.
String INQUIRY_ACK = "Found "; //String usada para identificar uma busca [por dispositivos] bem sucedida
String INQUIRY_NOT_FOUND = "No Devices Found"; //String usada para identifica uma busca sem resultados
const long INQUIRY_DELAY = 10000;

boolean debugOn = true; //Ativa/desativa o modo debug (Monitor Serial);
boolean echoOn = false; //Ativa/desativa o modo 'echo' dos comandos enviados e recebidos do bluetooth;

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

Figura 4.10 – Comandos de busca do Módulo Bluetooth

Outra função importante é a lastAuthorizedDevice. Esta função armazena o último dispositivo a acionar o motor. Ela é importante para garantir que o acionamento do motor não seja indevido.

Esta parte do código, ilustrada na figura 4.11, tem o objetivo organizar os dispositivos encontrados separados por um ponto e vírgula. Essa função também verifica se algum dos dispositivos encontrados está na lista de dispositivos autorizados. Já na figura 4.12, é feita a verificação para saber se o dispositivo foi o último a acionar o motor.

```
void loop() {
  //Procura dispositivos 'near by'
  String devices = inquiry();
  if(devices.length() > 0) {
    debug("Devices Found: " + devices);

    //Devices: 0023F1B2ADEA,Mari;00234DEA3DFD,2SP2BK1;
    int indexInicial = 0;
    int indexOfPontoVirgula = devices.indexOf(";");
    while(indexOfPontoVirgula > 0) {
      String device = devices.substring(indexInicial, indexOfPontoVirgula);
      debug("Device: " + device);
      if(isDeviceAuthorized(device) && shouldDeviceOpen(device)) {
        lastAuthorizedDevice = device;
        openCloseOperation();
        break;
      } else {
        debug("Not Authorized Device: " + device);
      }

      indexInicial = indexOfPontoVirgula + 1;
      indexOfPontoVirgula = devices.indexOf(";", indexInicial);
    }
  }
}
```

Figura 4.11 – Código para pesquisa e verificação de dispositivos

```
  int indexInicial = 0;
  int indexOfPontoVirgula = AUTHORIZED_DEVICES.indexOf(";");
  while(indexOfPontoVirgula > 0) {
    String authorizedDevice = AUTHORIZED_DEVICES.substring(indexInicial, indexOfPontoVirgula);
    if(authorizedDevice.equals(device)) {
      debug("Authorized Device Found: " + device);
      return true;
    }

    indexInicial = indexOfPontoVirgula + 1;
    indexOfPontoVirgula = AUTHORIZED_DEVICES.indexOf(";", indexInicial);
  }

  return false;
}

/**
 * Verifica se o dispositivo foi o último a abrir
 */
boolean shouldDeviceOpen(String device) {
  //Serial.print("  Last Open Time: ");Serial.println((millis() - lastOpenTime), DEC);
  if(device.equals(lastAuthorizedDevice) &&
      (millis() - lastOpenTime <= LAST_DEVICE_IGNORE_DELAY)) {
    debug("Device has opened the gate within less then 60 seconds");
    return false;
  }

  return true;
}
```

Figura 4.12 – Código para verificação dos dispositivos encontrados

4.7.3 – Código para o acionamento do motor, leitura dos sensores e controle do LED

Esta parte do código, ilustrada na figura 4.13, tem o objetivo executar a operação de abrir a porta deslizante, esperar o tempo necessário e fechar a porta. Simultaneamente a essa execução, o acionamento do LED na respectiva cor informará o estado do portão.

```
  //Armazena o último instante de operação
  lastOpenTime = millis();
}

void open() {
  debug("Open");
  analogWrite(LIGADO_PIN, VELOCIDADE_MOTOR);
  digitalWrite(FRENTE_PIN, HIGH);
  digitalWrite(TRAS_PIN, LOW);
}

void close() {
  debug("Close");
  analogWrite(LIGADO_PIN, VELOCIDADE_MOTOR);
  digitalWrite(FRENTE_PIN, LOW);
  digitalWrite(TRAS_PIN, HIGH);
}
```

Figura 4.13 – Código para acionamento do motor

Uma parte importante para o funcionamento correto do portão é a leitura dos sensores fim de curso. Enviado o digitalWrite(FRENTE_PIN, HIGH) e digitalWrite(TRAS_PIN, LOW) o portão irá se abrir.

Uma vez aberto, o portão e fica no loop enquanto não atingir o 'fim de curso' B, ABERTO ou enquanto não chegar no 'fim de curso' em 10 segundos. Se o tempo de operação tiver corrido e o portão ainda não tiver chegado ao sensor fim de curso ABERTO no final, então o LED ficará piscando azul intermitentemente até que o problema seja solucionado. Caso sensor B tenha sido acionado, o portão parará o Arduino, armazena o instante em que o portão abriu e fica no em um loop enquanto o tempo de abertura de 10 segundos não tiver passado. Nesse cenário, o LED iluminará verde. O LED ficará azul enquanto a porta deslizante estiver transladando. Depois de decorrido os 10 segundos de abertura, o Arduino envia novamente comandos digitalWrite(FRENTE_PIN, LOW) e digitalWrite(TRAS_PIN, HIGH) para o fechamento do portão. A figura 4.14 ilustra esse código.

```
void openCloseOperation() {
  debug("Open/Close Operation...");

  //Abre o portão e fica no loop enquanto não atingir o 'fim de curso'.
  open();
  lightYellow();

  //Fica em loop enquanto não chegar no 'fim de curso' ABERTO.
  long operationTimestamp = millis();
  while(!digitalRead(OPENED_SIGNAL)) {
    delay(10);
    //Se o tempo de operação tiver corrido e o portão ainda não tiver chegado
    //no final, então pisca amarelo intermitentemente
    if(millis() - operationTimestamp > OPERATION_DELAY) {
      stop();
      blinkYellow();
    }
  }

  //Pára o portão, armazena o instante em que o portão abriu e fica no
  //loop enquanto o tempo de abertura não tiver passado.
  stop();
  lightGreen();

  //Fica em loop enquanto não passar o tempo de ficar aberto.
  long openTimestamp = millis();
  while(millis() - openTimestamp < OPEN_DELAY) {
    delay(10);
  }

  //Fecha o portão e fica no loop enquanto não atingir o 'fim de curso'.
  close();
  lightYellow();
```

Figura 4.14 – Código para controle de leitura dos sensores

As funções do LED foram configuradas conforme a figura 4.15. A configuração do LED foi realizada a partir do tutorial de cores (LILYPAD, 2011)

```
void yellow() {
//  color(127,127,0);//AMARELO
  color(0,0,255);//AZUL
}

void red() {
  color(255,0,0);
}

void green() {
  color(0,255,0);
}

void lightsOut() {
  color(0,0,0);
}

// the color generating function
// fonte: http://web.media.mit.edu/~leah/LilyPad/06_rgb_code.html
void color (unsigned char red, unsigned char green, unsigned char blue)
{
  analogWrite(LED_VERMELHO, red);
  analogWrite(LED_AZUL, blue);
  analogWrite(LED_VERDE, green);
}

void debug(String msg) {
  if(debugOn) Serial.println(msg);
}
```

Figura 4.15 – Código para controle do LED RGB

CAPÍTULO 5 – TESTES E RESULTADOS

Neste capítulo são apresentados os testes e os resultados realizados para este projeto de pesquisa.

5.1 – Descrição das Etapas do Projeto

Como primeira etapa, a construção do protótipo de um portão por meio de um marceneiro experiente, foi realizada a partir de um esboço do portão. Neste esboço incluíam-se as exigências físicas nele detalhadas, de acordo com as necessidades do projeto. O encaixe para a passagem dos fios também foram incluídas nesta parte.

Já segunda etapa inclui a compra de materiais necessários para a montagem do projeto, tais como: dois breadboards de cores distintas, protoboard para testes, tubos de solda, o próprio ferro de solda, metros de fios coloridos e maleáveis, cabos *jumpers,* resistores de 10K e 220Ω, diferentes tipos de alicates e um estilete. Juntamente foram comprados os primeiros componentes eletrônicos para a confecção dos circuitos, como a placa Arduino, modem Bluetooth, MegaShield, três pares de sensores de fim de curso, motor DC reduzido, um CI L293D, e um LED RGB.

Para terceira etapa baseou-se na compra da fonte de alimentação de 9 volts demandada pelo motor utilizado no projeto. Em seguida foi desenvolvida a montagem do circuito do motor com o CI de ponte H em uma das breadboards para proporcionar o correto funcionamento do motor.

A quarta etapa da foi relacionada à montagem do MegaShield na placa Arduino Mega. Em seguida, foi a montagem do modem Bluetooth no segundo breadboard. Já para quinta etapa foi a realização das conexões do modem Bluetooth, com o CI e a placa Arduino.

5.2 – Testes

Neste tópico são relatados os principais testes efetuados para a implementação do sistema do projeto.

5.2.1 – Testes com o motor DC

O protótipo originou-se com a criação de uma porta deslizante com proporções menores para simular a abertura de um portão. O material escolhido para a porta foi uma madeira leve

confeccionada por um marceneiro. Uma vez pronta, a porta teve que ser ajustada para o encaixe de um trilho e o motor para a abertura e o fechamento da mesma. Para isso, foram utilizados dois trilhos de uma gravadora de DVD-ROM. Os trilhos foram cortados e colados na parte deslizante da porta. Com os trilhos posicionados, o motor foi encaixado em um apoio fixo aparte do portão, no entanto próximo a mesma. Este encaixe foi necessário para proporcionar ao motor maior estabilidade e permitir a simulação de erros na ausência ou mau encaixe de um motor. Uma vez encaixado a engrenagem do motor no trilho da porta, o deslizamento pode ser testado, conforme a figura 5.1.

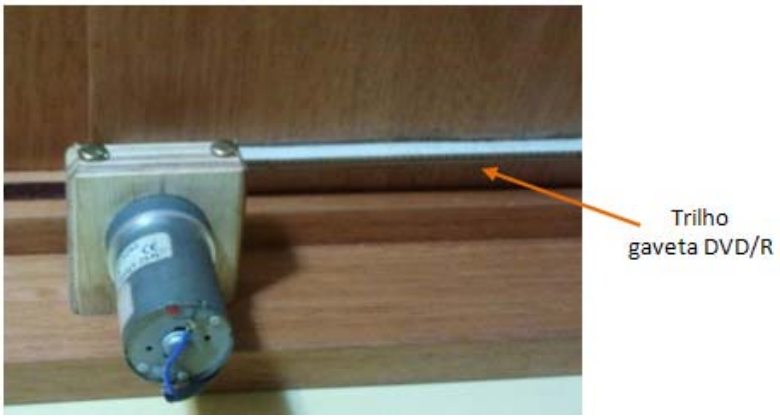


Figura 5.1 – Testes do motor DC fixado na janela.

Durante a montagem da porta, os encaixes para o LED e os sensores fim de curso também foram instalados na parte superior da porta. A figura 5.2 ilustra seu posicionamento e o encaixe de seus fios.

Figura 5.2 – LED e Sensores fim de curso fixados na moldura da porta.

Enquanto a porta estava sendo confeccionada, a montagem do circuito do motor com o CI de Ponte H, como mostrado na figura 5.3 foi iniciado para a realização do teste. Esta primeira versão do circuito foi realizada para certificar-se de que o circuito e a programação desenvolvida até aquele ponto estavam corretos.

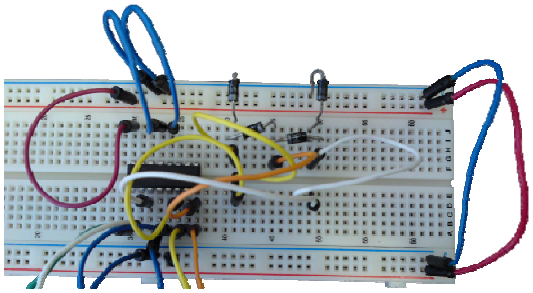


Figura 5.3 – Primeira versão do circuito construído para o motor DC.

Uma vez instalado o circuito de ponte H, como ilustrado na figura 5.4, foi possível realizar testes com o motor. Utilizando uma bateria de 9v como fonte de alimentação, este teste permitiu descobrir se o motor foi ligado corretamente com o CI e saber para que lado o motor estava girando. O teste obteve resultados positivos e foi possível controlar para que lado o motor girará.

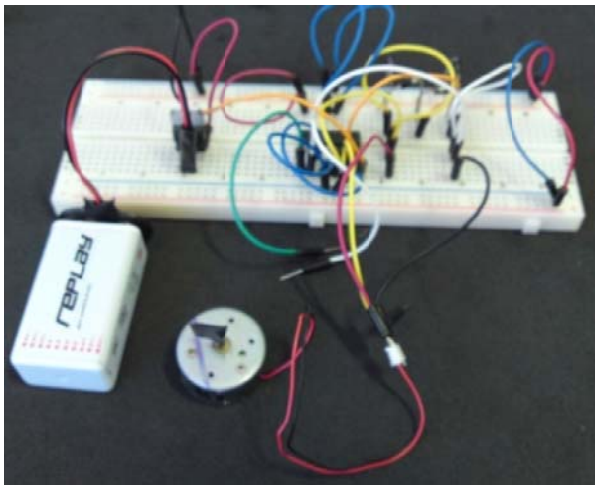


Figura 5.4 – Primeiro teste do circuito do motor DC.

Após a finalização dos testes com o motor DC iniciou-se a implementação do programa de pesquisa do módulo Bluetooth. Os primeiros testes feitos foram simplesmente para o acionamento do motor, independente do dispositivo Bluetooth, conforme a figura 5.5. Uma vez reconhecidos os dispositivos Bluetooth, o Arduino aciona o circuito do motor DC, permitindo a o giro do motor em sentindo horário para abertura e anti-horário para o fechamento da porta.

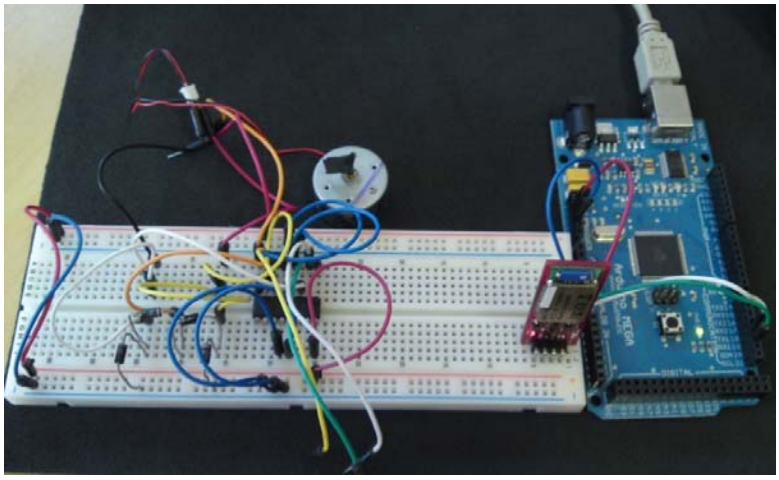


Figura 5.5 – Versão do circuito do motor DC acionado pelo modem

5.2.2 – Testes com as chaves fim de curso e o LED

Como a chave fim de curso utilizada neste projeto foi a *Magnetic Reed-Switch*, o acionamento das mesmas pode ser feita a certa distância. Sendo assim, uma vez acionado o motor reduz a velocidade até chegar à parada total. Desta maneira a porta deslizante não sofreria nenhum impacto ao chegar ao final de seu curso. A figura 5.6 ilustra o posicionamento dos sensores no protótipo.

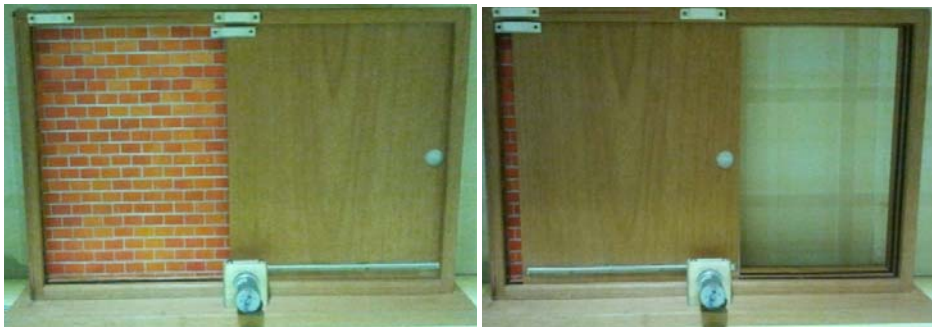


Figura 5.6 – Chaves *Magnetic Reed-Switch* coladas no protótipo.

Similarmente ao funcionamento, o LED RGB espera resposta do sensor fim de curso assim como também o comando do Arduino para demonstrar o estado da porta. Com o sensor funcionando corretamente, o acionamento do LED em teste foi simples e bem sucedida. A configuração deste acionamento teve com base no tutorial (LILYPAD, 2011).

5.2.3 – Testes com o módulo Bluetooth Mate e a Arduino Mega

Durante os testes com o modem, os resultados só poderiam ser visualizados no monitor serial do Arduino IDE. O primeiro teste foi realizado visando encontrar os endereço MAC e o nome dado ao dispositivo pelo usuário. O comando I<time>,<CoD> realiza uma busca padrão de 10 segundos e no máximo 48. CoD, *Code of Device,* significa código de dispositivo. Este comando então retorna até nove dispositivos por pesquisa no formato abaixo:

<BT address>,<BT name>,<COD>

00A053000164,MinhaPorta,72010C

Neste primeiro teste também foi cogitada a ideia de gravar na memória EEPROM da placa Arduino os endereços. No então, devido a sua capacidade limitada de 4KB notou-se então que se utilizada à memória EEPROM, a placa Arduino Mega teria sua vida útil reduzida.

Diante de tal impasse foi utilizado então o comando IN<time>,<cod>. Este comando realiza uma pesquisa similar ao comando I mencionando anteriormente, no entanto não retornará o nome do dispositivo Bluetooth. Devido à simplicidade da pesquisa, o retorno é mais rápido já que para a obtenção do nome exige um controle remoto de pesquisa para cada

dispositivo encontrado. E como uns dos objetivos deste projeto são a praticidade e a comodidade, um retorno rápido é fundamental.

5.3 – Resultados

Após o desenvolvimento dos circuitos e da programação do software em funcionamento, os testes confirmam as teorias discutidas neste projeto. O protótipo foi desenvolvido em etapas e testado conforme o avanço dos resultados obtidos do mesmo. Os testes são essenciais para validar a integração das diferentes partes do projeto, evitando assim, problemas de funcionamento.

5.4 – Circuito Final no Arduino

Todos os circuitos já abordados individualmente foram acoplados em duas *breadboards* fixados sobre o MegaShield, encaixado no Arduino Mega. Os circuitos não foram soldados em uma placa, pois o objetivo deste trabalho é desenvolver um protótipo e não um produto final.

A figura 5.7 ilustra a conexão dos circuitos. Por meio de setas, a disposição dos componentes eletrônicos foi simplificada para suas principais conexões.

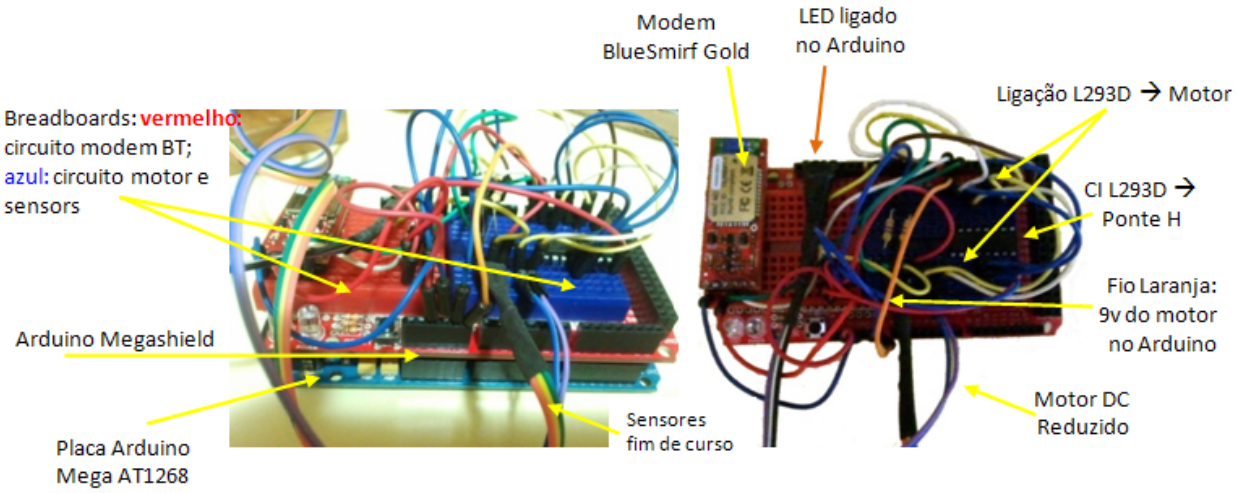


Figura 5.7 - Arduino com o circuito final completo.

No circuito apresentado acima, nota-se que todos os circuitos discutidos estão presentes nos *breadboard,* que em realidade são como mini protoboards. O breadboard vermelho possui as conexões relacionadas ao modem Bluetooth tanto para fonte de alimentação quanto para transmissão de dados. O modem Bluetooth está ligado na porta serial 3 da Arduino.

Referente ao *breadboard* azul nota-se o controle dos sensores, LED, ponte H e o motor DC, assim como também a fonte de alimentação de cada. A fonte de alimentação de 9v do motor, representada pelo fio laranja, está conectado ao pino VIN da Arduino. E por fim, CI de Ponte-H está ligado nas portas 2, 3 e 4 da Arduino. A porta 2 destina-se a controlar a velocidade do motor, que recebe o output analógico de 255. Enquanto a porta 3, ligada pelo fio branco, e a porta 4, ligada pelo fio amarelo, destinam-se ao INPUT 1 e 2 da Arduino responsável por inverter a polaridade do motor para girar do lado correto.

LED RGB está ligado com comum numa resistência de 220Ω para o GND com intuito de reduzir ruídos. O fio branco está conectado à porta PWM 13 da Arduino e com o pino vermelho do LED. Já o fio roxo está na porta PWM 11 conectada ao pino verde. E por fim o fio cinza corresponde à conexão do pino azul ligado a porta PWM 12.

Similarmente, os dois sensores fim de curso estão com uma resistência de 470Ω também para GND para remoção de ruído na Arduino. O fio laranja dos sensores está ligado aos 5v da Arduino enquanto o fio amarelo faz conexão de um sensor com o terra e ao pino 6 da Arduino. Da mesma maneira o fio verde estabelece o mesmo tipo de conexão ao terra, no entanto conecta-se ao pino 5 na Arduino.

5.5 – Protótipo Final

Todas as tarefas propostas para a conclusão deste trabalho foram executadas com sucesso desde a montagem geral do protótipo. As figuras 5.8 e 5.9 mostram o protótipo do projeto montado e em pleno funcionamento. O projeto como um todo é composto pela porta deslizante, sensores, LED, motor DC e o circuito desenvolvido com a tecnologia Arduino.

Figura 5.8 - Arduino com o circuito final completo.

Os resultados obtidos com o protótipo demonstraram ser satisfatórios diante do fato que atingiram o objetivo proposto no projeto. Com o funcionamento de acordo com o esperado, o protótipo provou ter uma praticidade e comodidade diante do cadastramento e acionamento simples dos elementos necessários para seu funcionamento. Contudo, este protótipo abordado neste trabalho se limita a isto, deixando de considerar qualquer aspecto de segurança ou questões de custo/benefício que possam ser levantadas. A figura 5.9 ilustra o resultado da operação através do monitor do Arduino IDE.

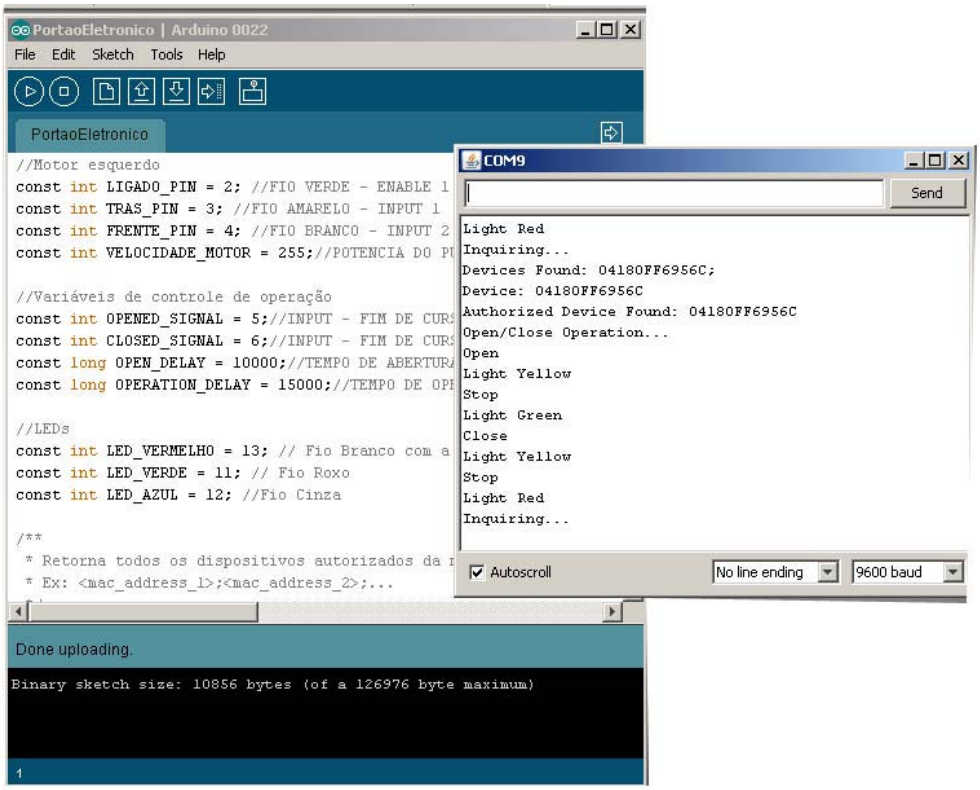


Figura 5.9 – Display das operações pelo Arduino IDE.

O protótipo do portão proporcionou uma simulação correta do acionamento do motor elétrico ativado desde a autorização do dispositivo Bluetooth do celular. De uma forma geral o sistema criado para esta simulação mostrou-se ser uma aplicação útil a tal tecnologia presente no celular. A figura 5.10 demonstra o protótipo em funcionamento.

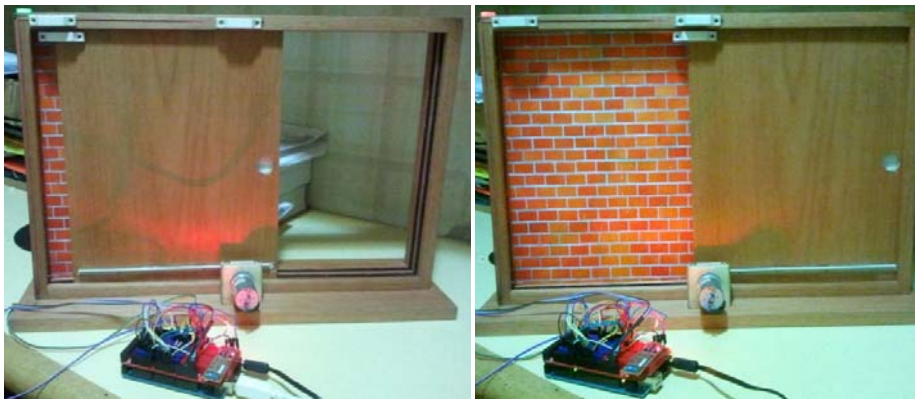


Figura 5.10 - Protótipo portão aberto e fechado.

5.6 – Avaliação Global do Projeto

O projeto apresentado tem potencialidade de utilização no mercado. Portões de chácaras, fazendas ou condomínios, que possuem um portão a certa distância da área residêncial, são ideais para a instalação do sistema. O sistema não só eliminaria a necessidade de vários controles como o portão já estaria aberto quando o usuário chegar próximo. Outra vantagem é que o LED indicará qualquer problema com o portão.

É bom observar que, para implementação deste projeto em portões reais em um edifício, residência ou condomínio, o motor teria que ser adaptado às condições do portão. Isso significa que portões ou catracas de grande porte necessitariam de uma maior potência para seu funcionamento. Nesse caso, o portão utilizaria motores mais potentes e os dispositivos eletrônicos teriam que ser adaptados às novas condições impostas.

5.7 – Dificuldades Encontradas

Durante todo desenvolvimento do projeto, foram encontradas diversas dificuldades que causaram um atraso na concepção do projeto.

5.7.1 – Programação

A aprendizagem e adaptação de novas linguagens de programação são sempre um desafio. Mesmo sendo baseado em *Wiring e Processing,* o Arduino possui uma linguagem de programação bem parecida com a linguagem C. No entanto, nem toda programação é idêntica a linguagem C. Devido suas peculiaridades, os comandos utilizados neste projeto tiveram que ser estudados cuidadosamente nos *datasheets* do modem Bluetooth e Arduino Mega. Outros tutoriais como o (LILYPAD, 2011) também foram utilizados para a configuração de controles como o do LED. A maior dificuldade encontrada foi fazer o correto uso dos comandos no tempo fornecido para obter os resultados desejados.

Para facilitar a programação, comandos de ação do módulo Bluetooth demonstrados no anexo foram utilizados. O código completo também pode ser encontrado no apêndice.

5.7.2 – Aquisição de dispositivos

Arduino é uma tecnologia que está em expansão devido à fácil adaptação em protótipo. No entanto, no Brasil há poucos lugares para aquisição de dispositivos eletrônicos. As placas, principalmente as Arduino, têm um custo não muito baixo e somente poucos sites como o RoboCore e Mercado Livre possuem tais dispositivos. Além do custo, o tempo de espera para a aquisição dos mesmos também não é imediata.

Para solução deste problema foi sugerido que a aquisição das peças seja em sites americanos como o Sparkfun. Em sites desse tipo são fornecidas uma maior variedade de dispositivos eletrônicos e Arduino. Além do bom preço, eles também realizam e aceitam entregas internacionais. A desvantagem é que por ser internacional, o tempo de espera é maior. Todavia, pode-se pagar mais caro por uma entrega mais rápida. Mesmo com esse custo a mais, as compras nos sites americanos ainda viabilizam a aquisição das mesmas. Sendo assim, com um bom planejamento as aquisições das peças podem ter um custo bem reduzido, viabilizando ainda mais a realização deste projeto.

CAPÍTULO 6 – CONCLUSÃO

Com o mundo se movimentando globalmente para a integração e automação de produtos, a sociedade que utiliza o computador e dispositivos do gênero sempre estão à procura de aplicações dos mesmos em seu cotidiano. Diminuir a subutilização de recurso presentes resulta em maior praticidade e comodidade para os usuários dessas tecnologias.

O projeto proposto foi desenvolvido especificamente para simular o melhor aproveitamento da tecnologia Bluetooth presente nos dispositivos móveis como o celular. É importante observar que essa tecnologia está presente hoje em dia até em celulares que não são da linha dos *smartphones,* o que apresenta ser mais uma vantagem para a aplicação proposta.

O sistema desenvolvido para a simulação da aplicação do acionamento do motor oferece certo controle de acesso, uma vez que os dispositivos têm que ser cadastrados anteriormente ao uso. A pesquisa de dispositivos permite identificar possíveis usuários que estão nas proximidades, facilitando também o cadastro de futuros usuários.

Para a realização do projeto, foi necessária a criação de um *hardware* e de um *software,* para a simulação da aplicação proposta. Para alcançar os objetivos discutidos, foi utilizada uma placa Arduino com microcontrolador AT1280 e um modem Bluetooth responsável pela busca de usuários e interpretação dos dados recebidos pelo mesmo e outros dispositivos. Foi utilizado um módulo Bluetooth acoplado em uma placa Arduino Mega para estabelecer a identificação do Bluetooth do dispositivo móvel, como o celular. A placa Arduino Mega, que foi programada, também foi utilizada para verificar os dispositivos Bluetooth presentes na área, assim como também identificar os dispositivos autorizados. Uma vez identificados e autorizados, o Arduino aciona motor DC, fazendo com que o portão abra e feche de acordo com a programação. Para que esse acionamento ocorra corretamente foi utilizado um circuito integrado de Ponte H, L293D, onde este CI também foi responsável por controlar a velocidade de rotação do motor.

Com a ajuda de sensores de fim de curso e um LED RGB, o sistema possibilitou uma maior interação e comodidade do usuário para saber o estado do portão. Dessa maneira, a ideia é que o usuário possa então tomar a providência necessária para caso ocorra algum problema de funcionamento do portão antes de estar em sua frente.

Para concluir, os resultados mostrados neste projeto comprovaram que as dificuldades encontradas foram superadas. Esses resultados também proporcionaram maior segurança, já

que o dispositivo tem que ser previamente cadastrado, praticidade, pois o usuário somente precisa levar consigo o celular, e conforto para seus usuários. A possibilidade de Com o protótipo funcionando conforme o esperado, é possível implementar a aplicação apresentada neste projeto em portões reais em um edifício, residência ou condomínio, adaptando somente o motor às condições do portão, já que portões utilizariam motores mais potentes e os dispositivos eletrônicos teriam que ser adaptados às novas condições impostas.

6.1 – Sugestões de Trabalhos Futuros

Como todos os sistemas existem aspectos que podem ser melhorados. Como sugestão para outros projetos, recomenda-se a criação de uma interface simples para o cadastramento e monitoramento dos usuários. O sistema atual requer a inserção do endereço MAC do dispositivo Bluetooth diretamente no código. Ou seja, para aplicações comerciais e residenciais, o ideal seria uma interface. Assim qualquer usuário pode fazer o controle do sistema.

A junção de outros sensores na porta deslizante, tais como o sensor movimento, e até o infravermelho para verificação da presença de um veículo também proporcionaram melhoras ao projeto. Qualquer aspecto de segurança também é uma ótima sugestão para trabalhos futuros. No modelo atual, o projeto restringiu-se apenas a identificar e autorizar o dispositivo que estiver ao alcance do modem. Com isso, a porta deslizante pode vir a abrir involuntariamente, permitindo então a entrada de algum intruso.

Similarmente a questão de segurança, com a abertura da porta com a simples passagem do dispositivo, o usuário pode estar próximo ao dispositivo, no entanto não deseja acioná-lo. A criação de algum sistema para evitar esta operação tornaria o sistema mais apropriado para a utilização do mesmo em prédios. A utilização de outras tecnologias também aumentam a segurança do sistema.

Outra boa sugestão seria obter a resposta do *status* do portão no próprio celular, utilizando também a tecnologia Bluetooth, assim como também receber por mensagem o controle de acesso do mesmo. Dessa maneira, o usuário teria a sua disposição todo acesso ao portão.

REFERÊNCIAS BIBLIOGRÁFICAS

AQUAHUB (2010). *How Float Switches Work*. Acesso em 31 de maio de 2011, disponível em AquaHub: http://www.aquahub.com/store/howfloatsw.html

ARDUINO, HARDWARE (2011). *Arduino*. Acesso em 25 de maio de 2011, disponível em: http://www.arduino.cc/en/Main/Hardware

ARDUINO, SOFTWARE (2011). *Arduino*. Acesso em 25 de maio de 2011, disponível em: http://www.arduino.cc/en/Main/Software

ARDUINO. (2011). *Arduino*. Acesso em 25 de maio de 2011, disponível em: www.arduino.cc

ATMEL. (2011). *AT1280 Datasheet*. Acesso em 29 de maio de 2011, disponível em Atmel Corporation: www.atmel.com/dyn/resources/prod_documents/doc2549.pdf. Página 7

BEBOP. (2011). *Arduino e Bluetooth*. Data da reportagem: 30 de maio de 2011. Acesso em 31 de maio de 2011, disponível no Blog BEBOP Computação criativa: http://bebop.cc/blog/

BRITES, F. G., & SANTOS, V. P. A. (2008). *Motor de Passo.* Projeto acadêmico em Engenharia de Telecomunicações da Escola de Engenharia do Centro Tecnológico da Universidade Federal Fluminense. Programa de Educação Tutorial. Grupo PET-Tele. Data do trabalho: julho de 2008. Niterói – RJ.

CEFETRIO. (2010). *Protocolo.* Acesso em 12 de maio de 2010. Disponível em: http://www.cefetrio.hpg.ig.com.br/ciencia_e_educacao/8/trabalhos/0200/wap/Image4.gif

COLOURIS, George. Dollimore, Jean. Kindberg, Tim. (2007) *Sistemas Distribuídos: Conceitos e Projeto.* Tradução João Tortello. Porto Alegre, Editora Bookman. 4ª Edição, 2007. Páginas 74, 110-111, 118-120.

DAWSON, BREET. (2011). *Understanding Gear Reduction*. Acesso em 29 de maio de 2011, disponível no Team DaVinci Robotics: http://www.teamdavinci.com/understanding_gear_reduction.htm

FITZGERALD, A. E., KINGSLEY, J. C., & UMANS, S. D. (2008). *Máquinas Elétricas: Com Introdução à Eletrônica de Potência* (6ª ed.). (A. Laschuk, Trad.) Porto Alegre, RS, Brasil: Bookman.

FRANCISCO, A. (2009). *Motores Eléctricos* (2ª ed.). Lisboa, Portugal: Lidel – Edições Técnicas/ Etep – Edições Técnicas e Profissionais/ Livrimpor – Livros Técnicos.

FRITZING. (2011). *About Fritzing*. Acesso em 03 de junho de 2011, disponível no Fritzing Org: http://fritzing.org/

HACK. (2011). *Arduino Hack-Day*. Acesso em 29 de maio de 2011, disponível no Blog Bebop: http://bebop.cc/blog/wp-content/uploads/2011/04/Arduino-Hack-Day.pdf

HAMBLEN, MATT. (2010). *Vendas anuais de aparelhos Bluetooth*. Data da reportagem: 13 de outubro de 2010. Acesso em 20 de maio de 2011, disponível no IDG News Service: http://idgnow.uol.com.br/computacao_pessoal/2010/10/13/vendas-anuais-de-aparelhos-Bluetooth-devem-chegar-a-2-bilhoes-ate-2013/

HILLS, PAUL. (2005). *Speed Controllers.* Data da reportagem: 05 de outubro de 2005. Acesso em 31 de maio de 2011, disponível em: http://homepages.which.net/~paul.hills/SpeedControl/SpeedControllersBody.html

HP. (2011). *Impressão a partir de um dispositivo compatível com Bluetooth*. Acesso em 20 de abril de 2011, disponível em HP: http://h10025.www1.hp.com/ewfrf/wc/document?lc=pt&dlc=pt&cc=br&docname=c00252816

IBOPE. (2009). *IBOPE Mídia*. Acesso em 24 de abril de 2010. Disponível em: http://www.ibope.com/consumidor/.

IKEDA, A. (2011). *Teclado Virtual Projeta Teclas na Superfície da Mesa*. Data da reportagem: 3 de março de 2011. Acesso em 15 de abril de 2011, disponível em UOL Tecnologia: http://tecnologia.uol.com.br/ultnot/multi/2011/03/03/04021C3360D4996327.jhtm?teclado-virtual-projeta-teclas-na-superfcie-da-mesa-04021C3360D4996327

KINMORE. (2011). *Motor DC 12v com Redução*. Acesso em 01 de março de 2011. Disponível em: http://kinmore.en.alibaba.com/product/456432720-212371475/12V_Motor_with_Gear_Gear_Reducer.html

LED. (2011). *LED RBG*. Acesso em 03 de junho de 2011, disponível em China Young Sun LED Technology: http://www.sparkfun.com/datasheets/Components/YSL-R596CR3G4B5C-C10.pdf

LILYPAD, T. (2011). *Colors Tutorial*. Acesso em 03 de junho de 2011, disponível em: http://web.media.mit.edu/~leah/LilyPad/06_rgb_code.html

LIPOVETSKY, Gilles. (2008). *A felicidade paradoxal – Ensaio sobre a sociedade de hiperconsumo.* São Paulo, Companhia das Letras, 2008.

MEGA. (2011). *Arduino Mega.* Acesso em 23 de maio de 2011, disponível no site oficial Arduino: http://arduino.cc/en/Main/ArduinoBoardMega

MELLO, A. CAROLINA. (2010). *GPS com Bluetooth.* Data da reportagem: 07 de outubro de 2010. Acesso em 20 de maio de 2011, disponível no UOL Tecnologia: http://tecnologia.uol.com.br/guia-produtos/todos/2010/10/07/com-recursos-limitados-o-motonav-tn30-e-gps-compacto-e-facil-de-usar.jhtm

MILLER, Michael. (2001). *Descobrindo Bluetooth***.** Tradução Altair Dias Caldas de Moraes e Cláudio Belleza Dias. Rio de Janeiro, Editora Campus Ltda. 2001. Páginas 9, 27, 152.

NIELSON. (2011). *Celulares em alta no Brasil*. Data da reportagem: setembro de 2009. Acesso em 20 de maio de 2011, disponível em Nielson: http://br.nielsen.com/news/Celulares_em_alta_set09.shtml

OLHAR DIGITAL. (2011). *Um Espelho Retrovisor com GPS, Bluetooth e Jogos TouchScreen*. Data da reportagem: 27 de abril de 2011. Acesso em 20 de maio de 2011, disponível em Olhar Digital: http://olhardigital.uol.com.br/produtos/digital_news/noticias/espelho_retrovisor_com_gps_Bluetooth_e_jogos_touchscreen

PAZOS, F. (2002). *Automação de Sistemas e Robótica* (1ª ed.). Rio de Janeiro, RJ, Brasil: Axcel Books do Brasil.

PRESS. (2009). *IBOPE Mídia.* Acesso em 24 de abril de 2010. Disponível em: http://www.ibope.com/conectmidia/estudo/index.html

REED RELAYS AND ELECTRONICS. (2009). *FAQ on Reed-Switches and Reed Sensors*. Acesso em 14 abril de 2011, disponível em Reed-switch Reability: http://www.reed-sensor.com/Notes/General_Reed_Switch_Theory.htm

REED-SWITCH INFO. (2011). *Reed-Switch Info*. Acesso em 17 de abril de 2011, disponível em Reed-Switch Info: http://www.reed-switch-info.com/

SBRC. (2001).Cordeiro, Carlos de M. Sadok, Djamel F. H. *Avaliação de Desempenho de Redes Bluetooth usando o Modelo de Captura.* Página 775-790. Acesso em 22 de maio de 2011. Disponível em: http://www.lbd.dcc.ufmg.br:8080/colecoes/sbrc/2001/050.pdf.

SIG. (2010). Bluetooth SIG, *Bluetooth Special Interest Group*. Acesso em 10 de maio de 2010, Disponível em: http://www.Bluetooth.com/ENGLISH/TECHNOLOGY/Pages/default.aspx.

SOARES, Luiz Fernando Gomes. Lemos, Guido. Colcher, Sérgio. (1995). *Redes de Computadores: das LANs, MANs e WANs às Redes ATM.* Rio de Janeiro, Elsevier Editora Ltda. 2ª Edição, 1995. Páginas 102-106.

SOUSA, Maxuel Barbosa. (2002). *Wireless: Sistemas de Rede sem Fio.* Editora Alta Books Brasport Livros e Multimídia Ltda., 2002. Página 5.

SPARKFUN. (2011). *MegaShield*. Acesso em 25 de maio de 2011. Disponível em: http://www.sparkfun.com/products/9346

TANENBAUM, Andrew S. (2003). R*edes de Computadores.* Tradução Vandenberg D. de Souza. Rio de Janeiro, Editora Campus/Elsevier. 4ª Edição, 2003. Páginas 22, 311-322, 330-338.

THE CELLUON. (2010). *Evo Mouse, the Evolution of the Mouse*. Data da reportagem: 27 de setembro de 2010. Acesso em 3 de março de 2011, disponível no Canal The Celluon: http://www.youtube.com/watch?v=UZWLwjbJZ-s

THOMPSON, SGS. (1996). *Datasheet L293D*. Acesso em 30 de maio de 2011, disponível em SGS-THOMSON Microelectronics: http://www.datasheetcatalog.org/datasheet/SGSThomsonMicroelectronics/mXyzuxsr.pdf

TORO, V. D. (1994). *Fundamentos de Máquinas Elétricas* (1ª ed.). (O. A. Martins, Trad.) Rio de Janeiro, RJ, Brasil: LTC.

UNICAMP. (2010). *Bluetooth – Características, protocolos e funcionamento.* Acesso em 12 de maio de 2010. Disponível em: http://www.ic.unicamp.br/~ducatte/mo401/1s2006/T2/057642-T.pdf

VINÍCIUS, S. (2008). *Fones Bluetooth têm poucas funções extras; Samsung pode ser controlado por voz.* Data da reportagem: 12 de agosto de 2008. Acesso em 3 de maio de 2011, disponível em UOL Tecnologia: http://tecnologia.uol.com.br/guia-produtos/audio/ult6183u5.jhtm

ANEXOS

A – Comandos de Ação do datasheet do módulo Bluetooth

4 Command Reference

All commands are either one or two characters and can be upper or lower case. Arguments for commands are delimited by a comma. Commands take decimal input except where noted. Text data, such as Bluetooth name, and pin code, are case sensitive. Commands fall into five general categories:

SET COMMANDS	store information to flash, changes take effect after power cycle or reboot
GET COMMANDS	retrieve and display the stored information
CHANGE COMMANDS	temporarily change the value of serial baudrate, parity, etc.
ACTION COMMANDS	perform action such as inquiry, connect, etc.
GPIO COMMANDS	configure and manipulate GPIO signals

1.2 SET Commands

All set commands do not take effect until after the module has been rebooted.

S7,<1,0> 7 bit data mode. 1 to enable, 0 to disable. (setting can be seen with the "d" command).

SA,<1,0> Authentication. 1 to enable, 0 to disable. This will force authentication when any remote device attempts to connect. Regardless of this setting, if a remote device forces authentication, this device will respond with the stored pin code. Once a remote device has exchanged pin codes with this device, a link key will be stored for future use. Up to 8 keys are automatically and permanently in flash on the device, in a first in, first out fashion.

SB,<timer> Send BREAK. This is an immediate command, which can send a BREAK signal on the TX . The timer is used to send a variable length BREAK signal.
Timer value Break length (in milliseconds)
1= 37ms, 2=18.5ms, 3=12ms, 4=9ms, 5= 7ms, 6=6ms.

Example : *"SB,2"* sends a 18.5 millisecond break signal.

SC,<hex word> Service Class (four hex values, 11 used, this is used with Device Class command below to create the 24 bit Class of Device number. Note the service class is interpreted by the inquiring device to determine the service. To see a complete listing of available Bluetooth service classes check the Bluetooth SIG web site.

Example : *"SC,0002"*

SD,<hex word> Device Class (four hex values, major and minor in a 16 bit word, used with service class above)

Example : *"SD,8040"*

To set the Class of Device (COD) to 0x1F0123 use the commands

SC,001F
SD,0123

SE,<1,0> Encryption 1 to enable, 0 to disable.

SF,1 Set Factory Defaults.

SI, <hex word> - Inquiry Scan Window. Sets amount of time device spends enabling inquiry scan (discoverability). Minimum value is 0x0012, corresponding to about 1% duty cycle. Inquiry interval is fixed at 0x800, so time spent in inquiry is 0x12/0x100 by default. Maximum value is 0x800, set to 0x0000 to disable inquiry scan and make device non-discoverable. Default value is 0x0200.

SJ, <hex word> Page Scan Window. Sets amount of time device spends enabling page scan (connectability) . Minimum value is 0x0012, corresponding to about 1% duty cycle. Page Scan interval is fixed at 0x800, so time spent in page scan mode is 0x12/0x800 by default. Maximum value is 0x800, set to 0x0000 to disable page scan and make device non-connectable. Default value is 0x0200.

SL,<E,O,N> Set UART parity. Can be any of, Even, Odd, or None. Only the first character is needed and must be capital.

Example : *"SL,E"* sets the parity to Even.

SM,<5,4,3,2,1,0> Mode (0=slave, 1=master,2=trigger, 3=auto, 4=DTR, 5=ANY)

Example : *"SM,1"* sets the mode to Master

SN,<name> Name of the device, 20 characters maximum.

Example: *"SN,MyDevice"*

S-,<name> Serialized Friendly Name of the device, 15 characters maximum.
This command will automatically append the last 2 bytes of the BT MAC address to the name. Useful for generating a custom name with unique numbering.

Example: *S-,MyDevice* will set the name to "MyDevice-ABCD"

SO,<text> Extended Status String, 8 character maximum. Setting this string to from 1 to 8 characters will enable status messages to be sent to the local serial port. Two status messages are sent, when a Bluetooth connection is established, the string "<text>CONNECT" will be sent. Upon a Disconnect, the string <text>DISCONNECT will be sent. This parameter is useful, for example, when connected to a printer, the printer can examine an escape sequence, if the <text> is set to ESC%, the printer can parse the ESC%CONNECT and ESC%DISCONNECT messages without interfering with normal print jobs. In Trigger or Master modes, the first character of this string is used as the BREAK connection character.

Example: *SO,ESC%*

SP,<text> Security pin code, 20 character maximum. Each time the device success pairs, the BT address will be saved. Up to eight addresses can be stored on a first in first out bases. To erase all stored pairings, reset the passkey command. You can use the same value that is already set.

Example: *SP,secretcode* sets pin code to "secretcode"

SQ,<num> Special configuration commands, num is a decimal number with the following interpretation.

Command	Description
0	Disable all special commands
4	Disable reading the values of GPIO3 and GPIO6 on power-up. This command is used when reprogramming GPIO3 and GPIO6 from their default configuration.
16	Configures the firmware to optimize for low latency data transfers rather than throughput.
128	Allow for fast reconnect. This allows applications to disconnect and reconnect back to back quickly.
256	Set 2 stop bit mode on the UART.

SR,<address> Store remote address, 12 hex digits, (6 bytes) no spaces or characters between digits

Example: *SR,00A053112233* sets the remote Bluetooth address to *00A053112233*

NOTE there are two special characters that can be used for the address parameter:

- SR,Z will erase any stored address.
- SR,I will write the last address seen using the inquiry command. This can be helpful when you just have only one other device in range.

SS,<text> Service Name (1 to 20 characters).

Example: *SS,SerialPort* service name set to "SerialPort"

ST,<number> Configuration timer, number of seconds (range= 0 to 255 decimal) to allow remote configuration over Bluetooth after power up in Slave Mode. In all Master modes, the remote configuration timer is set to 0 (no remote configuration). In Trigger Master Mode, the configuration timer is used as an idle timer to break the connection after time expires with no characters being received.

Examples:

ST,0 disables remote configuration
ST,60 sets remote configuration to 60 seconds (default value)
ST,255 enables remote configuration forever

SU,<rate> Baudrate, {1200, 2400, 4800, 9600, 19.2, 28.8, 38.4, 57.6, 115K, 230K, 460K, 921K }, only the first 2 characters are needed.

Example: *SU,57* sets the baudrate to 57600 baud.

SW,<hex word> Enable low power SNIFF mode. Default is 0000=disabled. SNIFF mode allows extreme low power operation. Device goes into a deep sleep, and wakes up every 625us * <hex word> to send/receive chars.

Example: *SW,0050* enables Sniff mode with interval time of 50 milliseconds

This will cause the module to enter low power sleep, and wake once every 50 milliseconds to check for RF activity. See Section 5.2 for more details on Sniff and managing power.

SX,<1,0> Bonding, if enabled only accept connections from the device that matches the stored Bluetooth address register will be accepted. The stored address register can be set with the SR command or will be set upon the first device pairing.

SZ,<num> Raw baud rate (decimal) allows entering of non-standard baud rates. Based on the formula num = baudrate * 0.004096.

S~,<0,1> Set profile to use according to the table below. See section 6.1 for more details on profiles.

<num>	Profile	Comments
0	SPP	Default , no modem control
1	DUN -DCE	Slave or gateway
2	DUN-DTE	Master or client
3	MDM SPP	With modem control signals
4	SPP and DUN-DCE	Multi-profile

S?,<0,1> Role Switch. Enables and disables Role Switch. If set, when an incoming connection is occurs to a slave mode device, an attempt will be made to force a role switch, allowing the slave to become the master. This is useful in situations where high speed data is being sent from the local device up to the remote host, and can result in better performance. However this may create a situation whereby the connecting host will not be able to make additional outbound connections (multipoint) while connected to this device. Default is DISABLED.

S$,<char> Configuration detect character. This allows a change from the default $$$ to some other character. Factory defaults returns the device to $$$.

Sl,<value> Low power connect mode. Disables the Bluetooth radio and LED timers while not connected. When set, the module will cycle between active (discoverable and connectable) and low power deep sleep. This can save considerable power when the module is waiting for long periods of time without a connection. The trade off is additional latency when connecting or pairing. The value is a four digit number

made up of two one byte intervals. The first interval is the ON period and the second the OFF period. Both are in seconds. The maximum value is 20 seconds for either of the periods. Default is 0000 always actively waiting for a connection.

Example: S|,0120 // creates a 5% duty cycle ON for one second and OFF for 20.

4.1 GET Commands

D Display basic settings. Address, Name, Uart Settings, Security, Pin code, Bonding, Remote Address.

E Display extended settings: service name, service class, device class, configuration Timer.

M Display remote side modem signal status.

O Display other settings. Configuration character, I/O port values, debug mode.

G<X> Display stored settings for command X. These commands correspond to the SET commands above.

Example: *GS* will return 1 or 0 depending on the value of security.

In addition to the above, there are a few other useful commands available**.**

GB Returns the Bluetooth Address of the device.

GK Returns the current connection status: 1=connected, 0 = not connected.

G& Return a hex byte containing the value of the PIO pins

V Return the software release version

4.2 CHANGE Commands

U,<rate>,<E,O,N> - Temporary UART Change, will change the serial parameters immediately, but not store them. Command will return "AOK" at current settings, then automatically exit command mode, and switch to new baud rate.

Baud rrate must be EXACTLY 4 characters = {1200, 2400, 4800, 9600, 19.2K, 38.4K, 57.6K, 115K, 230K, 460K, 921K }.

Parity is E, O, or N (must be capitals)

This command is effective immediately, does not require reboot.

Example: *U,9600,E* Sets baudrate to 9600, parity even.

4.3 ACTION Commands

$$$	Enter command mode Characters are passed as data until this exact sequence is seen. If any bytes are seen before or after the $$$ characters in a 1 second window, command mode will not be entered and these bytes will be passed on to other side. **NOTE:** The device will only enter command mode if it is within the configuration timer window (60 seconds from power up by default). In master mode the configuration timer is set to zero. The character string to enter command mode is configurable using the *S$* command Use **---** exit command mode. Exit command mode. "END" will be displayed.
+	Local echo. Toggle local echo of RX chars in command mode. (default is off).
C	Attempt to connect to the REMOTE stored address.
C,<address>	Connect to the address specified in hex format. The address is also stored as the REMOTE address.
CF<address>	Connect and immediately go into FAST data mode. NOTE: you will not be able to enter command mode while connected. PIO6 can still be used to disconnect. Thus PIO6 should be held HIGH before sending this command, as lowering PIO6 will cause a disconnect.
CFI	Connect and immediately go into FAST data mode using the LAST address found from the Inquiry command. NOTE: you will not be able to enter command mode while connected. PIO6 can still be used to disconnect.
CFR	Connect and immediately go into FAST data mode using the REMOTE address. Similar to the C command but bypasses the configuration timer.
CT<address>,<timer>	Connect with TIMER. The device will NOT use or store the remote address, rather will make a connection to the <address> (REQUIRED). The device will automatically disconnect after 7 seconds if no data is seen from UART or BT. An optional timer value can be entered to change the timer. This value is in ¼ seconds. So for a 30 second timer, use 120 as the value. The maximum value is 255 (64 seconds)
F,1	Go into fast data mode, ends configuration immediately.
I<,time>,<COD>	Performs an inquiry scan. Default time is 10 seconds, maximum is 48. COD parameter is optional, 0 or no entry looks for all device classes. When entering a COD you must provide all six characters, i.e. 0040F0 for COD 0x40F0. A maximum

of 9 devices will be returned. As devices are found, they are displayed in the format below:

<BT address>,<BT name>,<COD>
00A053000123,MySerialPort,72010C

IN<time>,<cod> Performs an inquiry scan like the *I* command but does not return the Bluetooth name so it returns much faster since getting the name requires a remote lookup for each device found.

IS<time> Performs an inquiry scan, with a COD of **0x001F00**, which is the default COD for Roving Networks Serial adapters and modules.

IR<time> Performs an inquiry scan, with a COD of **0x0055AA**, which is the special COD used By Roving Networks Serial adapters and modules to enable "instant cable replacement".

H Help, will print out a list of commands and their basic syntax

K, Kill (disconnect) from the current connection**.** The characters KILL<cr><lf> will be echoed to the local UART once the connection is broken.

L Link Quality. Returns real-time streaming link quality values at 5Hz. Value returned is a two bytes separated by a comma. A value of "ff" is the highest value. The first byte is the current reading the second byte is the low water mark. Example output:

RSSI =ff,e6

P,<char> Passes thru any characters up to a CR or LF while in command mode.

Q Causes device to be non-discoverable and non-connectable (temporarily). Does not survive a power cycle or reset**. Used with the Z command below. Use the "W" command to re-enable.**
This command will return **"Quiet"** as a response**.**

To get the lowest power mode, first issue a Q, then a Z. Use the SNIFF settings to get lowest power while connected.

R,1 Forces a complete reboot of the device (similar to a power cycle).

T,<0,1> Pass receive data (from UART or BT) while in command mode. Returns (T=0 , T=1 based on input).

W Re-enables discovery and connection. This command reloads the stored value of the Inquiry and Page Window to re-enable. For example, to turn off Discovery but still allow connections, send an **"SI,0000"** command, and follow it with **"W"** command. This command returns **"Wake"** as a response.

Appendix B: Command Quick Reference

SET COMMANDS		FACTORY SETTING
S7,<1,0>	- 7 bit data mode enable/disable	0= disabled
SA,<1,0>	- Authentication enable/disable	0= disabled
SB,<timer>	- Send BREAK	Not Applicable
SC,<hex word>	- Service Class	0x0000= unknown
SD,<hex word>	- Device Class	0x1F00= undefined
SE,<1,0>	- Encryption enable/disable	0=disabled
SF,1	- Factory Defaults	
SI,<hex word>	- Inquiry Scan window	0x0200
SJ,<hex word>	- Page Scan window	0x0200
SL,<E,O,N>	- Parity	N=None
SM,<0,1,2,3,4,5>	- Mode (0=Slave,1=mstr,2=trig, 3=auto, 4=DTR, 5=ANY)	0=Slave
SN,<text>	- Name	FireFly-xxxx
SO,<text> string	- Connect/Disconnect Status String	NULL= no status
SP,<text>	- Pin Code	1234
SR,<adr>	- Remote Address (SR,Z to remove)	NONE SET
SS,<text>	- Service Name	SPP
ST,<num>	- Config Timer	60 seconds
SU,<rate>	- Baudrate	115K
SW,<hex>	- SNIFF rate	0x0000=disabled
SX,<1,0>	- Bonding	0=disabled
S~,<0-4>	- Profile setting 0=SPP, 1=DCE, 2=DTE, 3=MDM, 4=DUN&SPP	0 = SPP
SZ,<num>	- Raw Baudrate	
S?,<0,1>	- Enable /Disable Role Switch	0=disabled

GET, DISPLAY COMMANDS

D	- Basic Settings
E	- Extended Settings
O	- Other Settings
G<X>	- Stored setting
H	- Help
GB	- BT Address
GK	- Connection Status
G&	- I/O Ports
V	- Firmware version

ACTION COMMANDS

+	- Toggle local echo of RX chars in command mode.
C	- Connect immediate to stored remote address.
C,<address>	- Connect to address.
CF<address>	- Connect Fast mode to address.
CFR	- Connect Fast mode to stored remote address.
CT<address>,<timer>	- Connect, required address, optional disconnect timer in ¼ seconds.
F,1	- Enter Fast data mode, end configuration immediate.
L	- Toggle link quality readings.
I,<time>,<cod>	- Device Scan Inquiry, time in seconds, optional cod = class of device filter, 0=all
IN<time>,<cod>	- Device Scan Inquiry, returns NAMEs.
IS<time>	- Device Scan Inquiry, fixed cod=0x001F00 to find Roving devices.
IR<time>	- Device Scan Inquiry, fixed cod =0x0055AA to find instant cable pairs.
K,	- Kill (disconnect) from current connection
Q	- Turn off Discovery and Connectability
R,1	- Reboot
T,<0,1>	- Pass receive data (from uart or BT) while in command mode.
U,<rate>,<E,O,N>	- Temp Uart Change
&	- return the value of the DIP Switches
W	- Re-enable Discovery and Connectability.
Z	- Enter low power Sleep mode

APÊNDICE

A – Código fonte do Projeto inserido no Arduino Mega

```
/*************
 * Declaração das constantes
com a pinagem do Arduino *
 *************/

//Motor esquerdo
const int LIGADO_PIN = 2; //FIO VERDE - ENABLE 1 (PWM)
const int TRAS_PIN = 3; //FIO AMARELO - INPUT 1
const int FRENTE_PIN = 4; //FIO BRANCO - INPUT 2
const int VELOCIDADE_MOTOR = 255;//POTENCIA DO PWM - ENABLE 1

//Variáveis de controle de operação
const int OPENED_SIGNAL = 5;//INPUT - FIM DE CURSO - Fio verde
const int CLOSED_SIGNAL = 6;//INPUT - FIM DE CURSO-  Fio amarelo
const long OPEN_DELAY = 10000;//TEMPO DE ABERTURA DO PORTÃO
const long OPERATION_DELAY = 15000;//TEMPO DE OPERAÇÃO DO PORTÃO

//LEDs
const int LED_VERMELHO = 13; // Fio Branco com a resistência de 440 ohms
const int LED_VERDE = 11; // Fio Roxo
const int LED_AZUL = 12; //Fio Cinza

/**
 * Retorna todos os dispositivos autorizados da memória separados por ';'.
 * Ex: <mac_address_1>;<mac_address_2>;...
 */
```

```
const String AUTHORIZED_DEVICES =
"E0F84723E267;0023F1B2ADEA;BC47605475B3;04180FF6956C;2021A5278576;"; //
endereços MAC autorizados


//Armazena o último dispositivo que abriu o portão

String lastAuthorizedDevice;

long lastOpenTime;

const long LAST_DEVICE_IGNORE_DELAY = 60000;


//Comandos do Bluetooth

String COMMAND_MODE = "$$$"; //Comando para entrar em modo de comando

String DATA_MODE = "---"; //Comando para sair do modo de comando

String CR = "\r\n"; //Quebra de linha (Carriage Return)

String INQUIRY = "IN5"; //Comando para pesquisar os dispositivos em volta.

String INQUIRY_ACK = "Found "; //String usada para identificar uma busca (por
dispositivos) bem sucedida

String INQUIRY_NOT_FOUND = "No Devices Found"; //String usada para identifica uma
busca sem resultados

const long INQUIRY_DELAY = 10000;


boolean debugOn = true; //Ativa/desativa o modo debug (Monitor Serial);

boolean echoOn = false; //Ativa/desativa o modo 'echo' dos comandos enviados e recebidos do
Bluetooth;


/*************

 * Início de operação*

 *************/



void setup() {

  Serial.begin(9600); // Comunicação entre Arduino e o Bluetooth


  //Inicializacao do modulo Bluetooth
```

```
  Serial3.begin(57600); //Porta serial TX/RX3 TTL Estabelece comunicação atraves da serial 3
- realizada pelos pinos RXTX 3 - padrão 15 RX Comunication pin 15 fio Verde

  //Inicialização do motor
  pinMode(LIGADO_PIN, OUTPUT);
  pinMode(TRAS_PIN, OUTPUT);
  pinMode(FRENTE_PIN, OUTPUT);

  //Inicialização dos "fim de curso"
  pinMode(OPENED_SIGNAL, INPUT);
  pinMode(CLOSED_SIGNAL, INPUT);

  //Inicialização dos LEDs
  pinMode(LED_AZUL, OUTPUT);
  pinMode(LED_VERDE, OUTPUT);
  pinMode(LED_VERMELHO, OUTPUT);
  lightRed();
}

/*
 */
void loop() {
  //Procura dispositivos que estão próximos
  String devices = inquiry(); // função de pesquisa
  if(devices.length() > 0) {
    debug("Devices Found: " + devices); // imprime no Monitor serial os dispositivos
encontrados

    // separa os endereços MAC por “;” e os mostram no Monitor serial
    int indexInicial = 0;
    int indexOfPontoVirgula = devices.indexOf(";");
    while(indexOfPontoVirgula > 0) {
```

```
      String device = devices.substring(indexInicial, indexOfPontoVirgula);
      debug("Device: " + device);
      if(isDeviceAuthorized(device) && shouldDeviceOpen(device)) {
        lastAuthorizedDevice = device;
        openCloseOperation();
        break;
      } else {
        debug("Not Authorized Device: " + device); // imprime no monitor serial os endereços MAC não autorizados
      }
// acha o ";" e vai para o próximo endereço. Utilizado para identificar os dispositivos
      indexInicial = indexOfPontoVirgula + 1;
      indexOfPontoVirgula = devices.indexOf(";", indexInicial);
    }
  }
}

/**
 * Verifica se algum dos dispositivos encontrados está na lista de dispositivos autorizados. Faz a conferencia só dos endereços ignorando os ";"
 */
boolean isDeviceAuthorized(String device) {
  int indexInicial = 0;
  int indexOfPontoVirgula = AUTHORIZED_DEVICES.indexOf(";");
  while(indexOfPontoVirgula > 0) {
    String authorizedDevice = AUTHORIZED_DEVICES.substring(indexInicial, indexOfPontoVirgula);
    if(authorizedDevice.equals(device)) {
      debug("Authorized Device Found: " + device);
      return true;
    }
```

```
    indexInicial = indexOfPontoVirgula + 1;
    indexOfPontoVirgula = AUTHORIZED_DEVICES.indexOf(";", indexInicial);
  }

  return false;
}

/**
 * Verifica se o dispositivo foi o último a abrir
 */
boolean shouldDeviceOpen(String device) {
  //Serial.print("  Last Open Time: ");Serial.println((millis() - lastOpenTime), DEC);
  if(device.equals(lastAuthorizedDevice) &&
     (millis() - lastOpenTime <= LAST_DEVICE_IGNORE_DELAY)) {
    debug("Device has opened the gate within less then 60 seconds");
    return false;
  }

  return true;
}

/*************
 * Bluetooth *
 *************/

/**
 * Faz a busca por dispositivos Bluetooth
 */
String inquiry() {
  debug("Inquiring...");
  commandMode();
```

```
  cleanBuffer();

  sendCommand(INQUIRY, true);
  delay(INQUIRY_DELAY); //Aguarda o retorno do INQUIRY

  String retorno = readResponse();
  //Serial.println("Retorno: " + retorno);

  //Se encontrar a string de "Found", faz o parse do
  //retorno para extrair os dispositivos encontrados
  if(retorno.indexOf(INQUIRY_NOT_FOUND) < 0 && retorno.indexOf(INQUIRY_ACK) >
0) {
    return parseInquiryReturn(retorno);
  }

  return "";
}

/**
 * Trata o retorno da busca dos dispositivos
 */
String parseInquiryReturn(String inquiryReturn) {
  String retorno = inquiryReturn;

  //Extrai do retorno apenas os dispositivos encontrados retirando a quebra de linha
  //Ex:
  //  Inquiry, COD=0
  //  E0F84723E267,38010C
  //  Found 1
  int index = retorno.indexOf("COD=0");
  index = retorno.indexOf(CR, index) + 2;
```

```
  int lastIndex = retorno.lastIndexOf(INQUIRY_ACK) - 2;
  retorno = retorno.substring(index, lastIndex); //Ex:
0023F1B2ADEA,23890F\r\nE0F84723E267,38010C


  //Substitui a quebra de linha entre os dispositivos por ';'
  retorno = retorno.replace(CR, ";") + ";";


  //Remove os CoD de todos os dispositivos, deixando apenas <mac_address>
  int indexOfPontoVirgula = retorno.indexOf(";");
//0023F1B2ADEA,5A0204;BC47605475B3,5A020C;00234DEA3DFD,3E010C;
  while(indexOfPontoVirgula > 0) {
    int lastIndexOfVirgula = retorno.lastIndexOf(",", indexOfPontoVirgula);//procura a primeira
vírgula antes do ';'
    retorno = retorno.substring(0, lastIndexOfVirgula) +
retorno.substring(indexOfPontoVirgula);


    indexOfPontoVirgula = retorno.indexOf(";", indexOfPontoVirgula);
  }


  return retorno;
}


/**
 * Coloca o dispositivo em modo de comando (se ainda não estiver)
 */
void commandMode() {
  //Limpa o buffer
  cleanBuffer();


  //Verifica se já está em modo de comando:
  //Para tal, envia um comando qualquer (existente). Se estiver em modo de comando haverá
resposta.
  sendCommand("GK", true); // Mostra o estado da conecção
```

```
  String retorno = readResponse();
  if(retorno.trim().length() > 0) {
    return;
  } else {
    //Se não tiver resposta, entra em modo de comando
    sendCommand("$$$", false);
    cleanBuffer();
  }
}

/**
 * Sai do modo de comando
 */
void dataMode() {
  sendCommand("---", true);
}

/**
 * Envia um comando para o Bluetooth.
 * Deve estar em modo de comando
 */
void sendCommand(String command, boolean cr) {
  if(cr) {
    Serial3.println(command);
    if(echoOn) Serial.println(command);
  } else {
    Serial3.print(command);
    if(echoOn) Serial.print(command);
  }
  delay(500);
}
```

```
/**
 * Lê a resposta do Bluetooth
 */
String readResponse() {
  String resposta = "";
  while(Serial3.available() > 0) {
    char retorno = Serial3.read();
    resposta += retorno;
  }

  if(echoOn && resposta.length() > 0) Serial.println(resposta);

  return resposta;
}

/**
 * Limpa o buffer do Bluetooth
 */
void cleanBuffer() {
  readResponse();
}

/************************
 * Controle dos Motores
 ************************/

/**
 * Executa a operação de abrir, esperar o tempo necessário e fechar o portão
 */
void openCloseOperation() {
```

```
debug("Open/Close Operation...");

//Abre o portão e fica no loop enquanto não atingir o 'fim de curso'.
open();
lightYellow();

//Fica em loop enquanto não chegar no 'fim de curso' ABERTO.
long operationTimestamp = millis();
while(!digitalRead(OPENED_SIGNAL)) {
  delay(10);
  //Se o tempo de operação tiver corrido e o portão ainda não tiver chegado
  //no final, então pisca amarelo intermitentemente
  if(millis() - operationTimestamp > OPERATION_DELAY) {
    stop();
    blinkYellow();
  }
}

//Pára o portão, armazena o instante em que o portão abriu e fica no
//loop enquanto o tempo de abertura não tiver passado.
stop();
lightGreen();

//Fica em loop enquanto não passar o tempo de ficar aberto.
long openTimestamp = millis();
while(millis() - openTimestamp < OPEN_DELAY) {
  delay(10);
}

//Fecha o portão e fica no loop enquanto não atingir o 'fim de curso'.
close();
```

```
  lightYellow();

  //Fica em loop enquanto não chegar no 'fim de curso' FECHADO.
  operationTimestamp = millis();
  while(!digitalRead(CLOSED_SIGNAL)) {
    delay(10);
    //Se o tempo de operação tiver corrido e o portão ainda não tiver chegado
    //no final, então pisca amarelo intermitentemente
    if(millis() - operationTimestamp > OPERATION_DELAY) {
      stop();
      blinkYellow();
    }
  }

  //Pára o portão
  stop();
  lightRed();

  //Armazena o último instante de operação
  lastOpenTime = millis();
}

void open() {
  debug("Open");
  analogWrite(LIGADO_PIN, VELOCIDADE_MOTOR);
  digitalWrite(FRENTE_PIN, HIGH);
  digitalWrite(TRAS_PIN, LOW);
}

void close() {
  debug("Close");
```

```
  analogWrite(LIGADO_PIN, VELOCIDADE_MOTOR);
  digitalWrite(FRENTE_PIN, LOW);
  digitalWrite(TRAS_PIN, HIGH);
}

void stop() {
  debug("Stop");
  digitalWrite(LIGADO_PIN, LOW);
  digitalWrite(FRENTE_PIN, LOW);
  digitalWrite(TRAS_PIN, LOW);
}

void blinkYellow() {
  //Fica em loop piscando o amarelo
  while(true) {
    debug("Blink Yellow");
    yellow();
    delay(250);
    lightsOut();
    delay(250);
  }
}

void lightYellow() {
  debug("Light Yellow");
  yellow();
}

void lightGreen() {
  if(debug) Serial.println("Light Green");
  green();
```

```
}

void lightRed() {
  debug("Light Red");
  red();
}

/**********************
 * Funções utilitárias *
 **********************/

void yellow() {
//  color(127,127,0);//AMARELO
  color(0,0,255);//AZUL
}

void red() {
  color(255,0,0);
}

void green() {
  color(0,255,0);
}

void lightsOut() {
  color(0,0,0);
}

// the color generating function
// fonte: http://web.media.mit.edu/~leah/LilyPad/06_rgb_code.html
void color (unsigned char red, unsigned char green, unsigned char blue)
```

```
{
  analogWrite(LED_VERMELHO, red);
  analogWrite(LED_AZUL, blue);
  analogWrite(LED_VERDE, green);
}

void debug(String msg) {
  if(debugOn) Serial.println(msg);
}
```