UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

DIEGO VANNUZINI PESSOA DE ALBUQUERQUE

DESENVOLVIMENTO DE UMA INTERFACE DE MANUTENÇÃO PARA DISPOSITIVOS EMBARCADOS UTILIZANDO *BLUETOOTH*

VITÓRIA – ES
DEZEMBRO/2014

DIEGO VANNUZINI PESSOA DE ALBUQUERQUE

DESENVOLVIMENTO DE UMA INTERFACE DE MANUTENÇÃO PARA DISPOSITIVOS EMBARCADOS UTILIZANDO *BLUETOOTH*

Parte manuscrita do Projeto de Graduação do aluno **Diego Vannuzini Pessoa de Albuquerque**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Orientador: Eng. MSc. Marcelo Souza Fassarella

Coorientador: Prof. Dr. Evandro Ottoni Teatini Salles

VITÓRIA – ES
DEZEMBRO/2014

DIEGO VANNUZINI PESSOA DE ALBUQUERQUE

DESENVOLVIMENTO DE UMA INTERFACE DE MANUTENÇÃO PARA DISPOSITIVOS EMBARCADOS UTILIZANDO *BLUETOOTH*

Parte manuscrita do Projeto de Graduação do aluno **Diego Vannuzini Pessoa de Albuquerque**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

COMISSÃO EXAMINADORA:

_____

Eng. MSc. Marcelo Souza Fassarella - Orientador

_____

Prof. Dr. Evandro Ottoni Teatini Salles - Coorientador
Universidade Federal do Espírito Santo

_____

Profa. Dra. Raquel Frizera Vassallo
Universidade Federal do Espírito Santo

_____

Prof. Dr. Anselmo Frizera Neto
Universidade Federal do Espírito Santo

Dedico aos meus pais, Luciane Vannuzini e Eduardo Pessoa, representando toda minha família.

Agradeço primeiramente a Deus, fonte de toda ciência que nos abençoa com o dom da vida.
Ao meu orientador Eng. MSc. Marcelo Souza Fassarella, à toda equipe do setor de engenharia da Gecontrol Ltda. e ao meu coorientador Prof. Dr. Evandro Ottoni Teatini Salles pelo conhecimento compartilhado. Ainda agradeço à própria Geocontrol Ltda pela disponibilização de material e espaço físico para o desenvolvimento do projeto.
A todos professores e funcionários do curso de Engenharia Elétrica da Universidade Federal do Espírito Santo por todo apoio e dedicação ao ensinar.
Aos meus familiares e amigos, em especial à Roberta, pela compreensão, apoio e incentivo.

RESUMO

A necessidade de mobilidade para que seja possível ter acesso a dispositivos instalados em campo despertou interesse em criar uma ferramenta eficiente e simples para que esse objetivo seja atingido. Em muitas situações, levar um notebook para campo se torna uma atividade penosa, porém um smartphone é um equipamento pequeno e que já está inserido no cotidiano de muitas pessoas. Sendo assim, este trabalho abordará o desenvolvimento de uma aplicação que possa ser executada em diversos modelos de smartphones que utilizem o sistema operacional *Android*. Além disso, a comunicação antes cabeada, com o uso de notebooks, agora passa a ser sem fio, utilizando a tecnologia *Bluetooth*, presente nestes aparelhos a serem utilizados. Logo, somado a isso será desenvolvido também um hardware que atue como intermediador na realização da comunicação. Este módulo se conectará ao equipamento final, ou seja o dispositivo que se deseja ter acesso, via cabo, e ao aplicativo *Android* pela tecnologia sem fio *Bluetooth* embarcada. Por fim, será apresentado o exemplo real implementado, em que se utilizará o equipamento rastreador (propriedade da Geocontrol Ltda.).

Palavras-chave: *Bluetooth*, *Android*, comunicação serial, manutenção, Eagle, MIT *App Inventor* 2 e Eclipse.

LISTA DE FIGURAS

Figura 1 – Hierarquia de protocolos *Bluetooth* 17
Figura 2 – Redes *piconet* e *scatternet* 20
Figura 3 – Diagrama de estados para dispositivos *Bluetooth* 21
Figura 4 – Tela de apresentação do Eclipse 24
Figura 5 – Módulo *Component Designer* do MIT *App Inventor* 25
Figura 6 - Módulo *Blocks Editor* do MIT *App Inventor* 26
Figura 7 – Diagrama do sistema objeto de estudo 27
Figura 8 – Módulo *Bluetooth*-UART HC-05 27
Figura 9 – Pinagem do módulo HC-05 28
Figura 10 – Esquemático do circuito eletrônico projetado 31
Figura 11 – Projeto final do hardware. 33
Figura 12 – Projeto simplificado 34
Figura 13 – Circuito impresso produzido 35
Figura 14 – Protótipo finalizado 35
Figura 15 - Diagrama do sistema objeto de estudo 36
Figura 16 – Cálculo exemplo do CRC 39
Figura 17 – Tela inicial do aplicativo 40
Figura 18 – Listagem das telas disponíveis 41
Figura 19 – Tela com campo em branco 42
Figura 20 – Conexão estabelecida 42
Figura 21 – Telas para envio de comandos 43
Figura 22 – Uso do campo de texto na tela da categoria configurador 44
Figura 23 - Rastreador 45
Figura 24 – Conectores Mini-Fit do rastreador 45
Figura 25 – Conexão entre o rastreador e o módulo *Bluetooth*-UART 46
Figura 26 – Tela do configurador após receber resposta de comando exemplo 47
Figura 27 – Foto do aplicativo em operação 47
Figura 28 – Tela com comando para configurar informações GPRS 48
Figura 29 – Foto do aplicativo na tela de envio do comando “Dados Operadora” 49
Figura 30 – Tela Analisador ao executar o comando “Debug GPS GGA” 50

LISTA DE QUADROS

Quadro 1 – Classificação das classes *Bluetooth* quanto ao alcance do enlace 16
Quadro 2 – Classes *Java* para acessar o *Bluetooth* do dispositivo *Android* 24
Quadro 3 – Pinagem detalhada do módulo HC-05 28
Quadro 4 – Comandos de configuração do módulo HC-05 29
Quadro 5 - Lista de componentes do hardware projetado 30
Quadro 6 – Lista de comandos disponíveis para configurar o rastreador..... 36
Quadro 7 – Consumo médio aproximado do hardware projetado 51

LISTA DE ABREVIATURAS E SIGLAS

CAN	*Controller Area Network* (Rede com Controlador de Área)
CRC	*Cyclic Redundancy Check* (Verificação de Redundância Cíclica)
CTM	Computador Tático Militar
GGA	*Global Positioning System Fix Data* (Dados do Sistema de Posicionamento Global)
GPS	Sistema de Posicionamento Global
GPRS	Serviço Gerais de Pacote por Rádios
GSV	*Satellites in view* (Satélites na Visada)
HCI	Interface de Controle de *Host*
IHM	Interface Homem Máquina
ISM	Industrial, Científica e Médica
L2CAP	Protocolo de Adaptação e Controle do Enlace Lógico
LCD	*Display* de Cristal Líquido
LMP	Protocolo de Gerenciamento de Enlace
MIT	Instituto de Tecnologia de Massachusetts
PAN	*Personal Area Networks* (Redes de Área Pessoal)
QoS	*Quality of Service* (Qualidade do Serviço)
RMC	*Recommended Minimum sentence C* (Mínima Sentença C Recomendada)
SMD	*Surface Mount Device* (Dispositivo de Montagem em Superfície)
UART	Transmissor/Receptor Universal Assíncrono
USART	Transmissor/Receptor Universal Síncrono Assíncrono
USB	Barramento Serial Universal

SUMÁRIO

1 INTRODUÇÃO 11
1.1 Apresentação 11
1.2 Objetivo geral e objetivos específicos 13
1.2.1 Objetivo geral 13
1.2.2 Objetivos Específicos 14
1.3 Estrutura do trabalho 14
2 TECNOLOGIA DE COMUNICAÇÃO SEM FIO BLUETOOTH 16
2.1 Arquitetura *Bluetooth* 16
2.1.1 Protocolo de Rádio 17
2.1.2 Protocolo de Baseband 18
2.1.3 Protocolo de LMP 18
2.1.4 Protocolo de HCI 19
2.1.5 Protocolo de L2CAP 19
2.2 Topologia da rede *Bluetooth* 20
2.3 Estados de um dispositivo *Bluetooth* 20
3 SISTEMA OPERACIONAL ANDROID 22
3.1 Interface de programação Eclipse 23
3.1.1 Ferramentas necessárias 23
3.1.2 Apresentação do Eclipse 23
3.1.3 Integração *Bluetooth* e *Android* 24
3.2 MIT *App Inventor* 2 25
4 PROJETO DO PROTÓTIPO BLUETOOTH – RS232 27
4.1 Módulo HC-05 27
4.1.1 Configuração do módulo HC-05 28
4.2 Projeto do hardware 30
5 DESENVOLVIMENTO EM PLATAFORMA ANDROID 36
5.1 Funcionalidades necessárias 36
5.2 Aplicativo desenvolvido utilizando a ferramenta Eclipse 38
5.3 Aplicativo desenvolvido utilizando ferramenta MIT *App Inventor* 2 39
5.3.1 Tela com campo em branco 41

5.3.2 Telas para envio de comandos.....43
6 CASO PRÁTICO – RASTREADOR.....45
7 CONCLUSÕES E SUGESTÃO DE TRABALHOS FUTUROS 52
8 REFERÊNCIAS BIBLIOGRÁFICAS 54
APÊNDICE A – ESQUEMÁTICO DO CIRCUITO ELETRÔNICO PROJETADO...57
APÊNDICE B – MAINACTIVITY.JAVA E ANDROIDMANIFEST.XML.....58

1 INTRODUÇÃO

1.1 Apresentação

Diante da grande quantidade de sistemas microcontrolados e embarcados existentes no mercado sendo usados para diversas finalidades, é de extrema importância que exista uma maneira eficiente de acessá-los. Assim, existem as interfaces UART/USART (Transmissor/Receptor Universal Assíncrono / Transmissor/Receptor Universal Síncrono Assíncrono) que são responsáveis pela transmissão e recepção de informação oriundas e destinadas a estes dispositivos. Essas interfaces são largamente utilizadas como ferramenta de manutenção, configuração e debug.

Além dessas interfaces descritas anteriormente, são utilizados também, com a mesma finalidade, o USB (Barramento Serial Universal) e a rede CAN (*Controller Area Network*), outros dois padrões de comunicação.

O acesso a estes dispositivos, na grande maioria dos casos, é feito através de cabos. Conectando a interface de acesso, na qual o operador recebe e envia, ao sistema embarcado. Porém, com o avanço da tecnologia, a comunicação sem fio é cada vez mais utilizada devido à mobilidade que proporciona. Assim, existem diversas tecnologias como *Wi-fi*, *Zigbee*, *Bluetooth*, *links* de rádio frequência, dentre outras. A primeira apresenta é mais utilizada em sistemas de múltiplo acesso doméstico e exige um consumo de energia mais elevado do que as tecnologias *Zigbee* e *Bluetooth*. Links de rádio frequência são utilizados em sua maior parte para comunicação de longas distâncias e potências de transmissão elevadas. As outras duas tecnologias citadas são utilizadas em rede pequenas e seu uso não gera um consumo de energia exacerbado. Porém *Zigbee* é mais utilizada em meio industrial ou de pesquisa não tão difundido entre a sociedade de maneira geral, que o caso do *Bluetooth*. Esta última tecnologia citada já é de uso difundido na sociedade pois está presente em diversos dispositivos, como celulares, computadores e *tablets*. Por este motivo, será alvo de estudo ao longo deste trabalho.

Um exemplo de aplicação desses sistemas embarcados citados, é o equipamento, conhecido como rastreador, desenvolvido na empresa Geocontrol Ltda. Este produto tem como principal funcionalidade registrar a posição geográfica (latitude e longitude) na qual um objeto a ser

rastreado (um veículo, por exemplo) se localiza ao longo do tempo. Seu funcionamento é simples, um módulo GPS (Sistema de Posicionamento Global) é embarcado no equipamento e tem a função de obter as coordenadas geográficas por onde este esteja. Este produto é capaz também de fornecer outras informações, como por exemplo, velocidade e sentido de movimento, dado que, ambas informações podem ser obtidas por este próprio módulo. Além do GPS, o rastreador possui um módulo GPRS (Serviço Gerais de Pacote por Rádios) que tem a função de transmitir estes dados via rede de dados móvel para um servidor onde os mesmos são armazenados e analisados.

Este rastreador é um produto comercializado por esta empresa e é utilizado em veículos como forma de melhorar a segurança do proprietário do veículo e prover serviços de interesse público. Por exemplo, pode ser utilizado para controlar o transporte público de uma grande cidade como Vitória.

Dado que estes equipamentos ficam instalados dentro de veículos, propositalmente em locais de acesso limitado, escondidos, a manutenção a ser realizada passa a ser uma tarefa difícil. Isso porque o técnico, ao prestar assistência, precisa levar um computador até o equipamento para acessá-lo com a finalidade de o reconfigurar e através de um cabo conectar o computador ao rastreador. Este procedimento, fisicamente, limita a operação pela distância do cabo. Além disso, é muito custoso para a empresa fornecer um computador portátil para cada técnico, logo este tipo de acesso acaba não sendo realizado com a frequência desejada.

Nesse exemplo, melhorar o acesso dos técnicos de manutenção aos rastreadores que já estão instalados pode ser de grande valia para a empresa. Portanto, fornecer mobilidade ao técnico pode consequentemente gerar agilidade no processo de manutenção destes dispositivos.

Seguindo uma abordagem similar à apresentada por Cruz (2013), a tecnologia *Bluetooth* atende às necessidades desse trabalho por realizar com êxito tal comunicação. Além disso, está presente em praticamente todo dispositivo móvel disponível no mercado, principalmente *tablets* e celulares. Vale ressaltar que é possível encontrar *tablets* e celulares com valor de compra inferior ao de um computador portátil. Este fator custo é algo determinante na escolha do projeto, pois é sempre mais vantajoso implementar um mesmo sistema utilizando menos recurso financeiro.

Logo, desenvolver um sistema que possa ser implementado em um aparelho celular e utilize *Bluetooth* para configurar os produtos desejados é justificado também pelo custo e pela alta disponibilidade no mercado. Além desses fatores já citados, a facilidade de uso e de acesso de um celular gera mais um ponto favorável a essa arquitetura.

Além desse produto citado, a empresa possui outros dispositivos que podem também ser acessados de maneira similar, como o Conecta (um sistema embarcado desenvolvido pela Geocontrol Ltda para fornecer acesso ao sistema online da polícia militar dentro de uma viatura) e o CTM (Computador Tático Militar). Dado que o rastreador é um sistema embarcado desenvolvido para servir de apoio aos policiais e se localiza nas viaturas e o CTM é um computador especificado para uso militar, resistente a quedas e a prova d'água, ambos podem ser acessados para fins de configuração ou manutenção.

Logo, fica claro que este trabalho pode ser útil para diversos equipamentos, pois qualquer sistema embarcado e/ou microcontrolado utiliza as interfaces já citadas.

1.2 Objetivo geral e objetivos específicos

Assim, este trabalho tem como objeto de pesquisa desenvolver uma nova maneira de realizar esta comunicação via tecnologia sem fio, facilitando o acesso a diversos sistemas remotos. Vale ressaltar, que este trabalho visa realizar esta comunicação sem alterar o produto original, tendo em vista que não é o objetivo desse desenvolvimento encarecer algo que já é consolidado no mercado. Simplesmente dinamizar processos já existentes. Para confirmar esta afirmação não será objeto de estudo deste trabalho um estudo de viabilidade da incorporação deste projeto ao equipamento já existente.

1.2.1 Objetivo geral

Acessar os rastreadores por meio de uma interface *Bluetooth*, com o objetivo de realizar diagnósticos e manutenção.

1.2.2 Objetivos Específicos

Desenvolver um aplicativo na plataforma *Android*, pois se trata de um sistema operacional de fácil acesso em muitos celulares e tablets comerciais, fornece uma interface amigável, a mobilidade desejada. Desenvolver também uma interface física que viabilize a comunicação entre o rastreador e o dispositivo *Android*. Vale ressaltar que ela precisa ser conectada ao rastreador sempre que for desejado acessá-lo e posteriormente desconectada quando o acesso for terminado.

1.3 Estrutura do trabalho

Este Capítulo 1 apresentou a temática deste trabalho, expondo os objetivos, a justificativa e as motivações que levaram a tratar desse assunto.

No Capítulo 2 será apresentada uma revisão com relação à tecnologia *Bluetooth*. Primeiramente uma visão geral dos aspectos físicos, como faixa de frequência, distância para viabilidade da comunicação e taxa de comunicação, será introduzida. Serão abordados, também, aspectos da arquitetura, como organização hierárquica de protocolos de comunicação e suas respectivas funcionalidades. Além disso, a topologia da rede e os possíveis estados em que os dispositivos podem estar dentro da rede serão citados.

No Capítulo 3 serão apresentados alguns aspectos essenciais da linguagem *Android* para que este projeto seja implementado. Além disso serão introduzidos os ambientes de programação utilizados no desenvolvimento do aplicativo proposto anteriormente neste Capítulo 1.

No Capítulo 4 será apresentado o projeto do hardware que foi desenvolvido com o objetivo de criar o enlace de comunicação entre o dispositivo móvel (*smartphone* ou *tablet*) e o equipamento rastreador a ser configurado. Para este fim foi utilizado o software *Eagle*.

No Capítulo 5, os aplicativos desenvolvidos para plataforma *Android* serão apresentados. Sendo um desenvolvido como a interface principal e outro como apoio ao funcionamento deste. O aplicativo principal viabilizou a interface gráfica necessária para que o profissional que realizará a manutenção no equipamento em campo tenha informações suficientes para

tomar suas decisões. E assim, realizar o procedimento correto para que o equipamento retome seu correto funcionamento.

O Capítulo 6 exemplificará a operação de manutenção que é o objetivo deste projeto. Assim, será apresentado o equipamento rastreador e então demonstrado um exemplo de funcionamento bem sucedido da comunicação entre o aplicativo desenvolvido e este equipamento.

No Capítulo 7 serão apresentadas as conclusões obtidas com este trabalho.

Na sequência, serão apresentadas as referências bibliográficas utilizadas para a elaboração deste trabalho

E por fim, serão apresentados os apêndices, contendo no Apêndice A, o esquemático do circuito eletrônico projetado e no Apêndice B, os códigos do aplicativo desenvolvido utilizando a ferramenta Eclipse.

2 TECNOLOGIA DE COMUNICAÇÃO SEM FIO BLUETOOTH

Conforme já definido anteriormente, a tecnologia utilizada para viabilizar a comunicação entre o celular e o rastreador será o *Bluetooth*. Esta tecnologia utiliza uma banda de radiofrequência chamada de ISM (Industrial, Científica e Médica), destinada para aplicações nesses ramos de atuação e é de uso livre. A faixa de frequência em questão vai de 2,4 GHz até 2,485 GHz. Este sistema utiliza uma técnica de espalhamento espectral por saltos em frequência, chegando a realizar 1600 saltos por segundo, para a transmissão dos dados. Essa técnica melhora a qualidade da transmissão. Através desse processo, é possível realizar a comunicação entre os usuários distantes de até 100 metros um do outro, formando as PAN's (*Personal Area Networks* – Redes de Área Pessoal) (*Bluetooth* SIG, a).

Esta técnica de espalhamento espectral foi desenvolvida com o objetivo de diminuir a interferência entre diferentes enlaces de comunicação. Assim, a banda anteriormente definida ficou dividida em 79 canais com largura de banda de 1MHz cada, o que gera um uso mais eficiente do espectro de frequência (*Bluetooth* SIG, b).

Foram definidas três classes de uso dessa tecnologia. Nelas se classificam a potência e o alcance do enlace. Vale ressaltar que, cada aplicação se encaixa na classe que atende aos seus anseios. No Quadro 1 é possível verificar tal classificação quanto ao alcance.

Quadro 1 – Classificação das classes *Bluetooth* quanto ao alcance do enlace

	Alcance
Classe1	100 metros
Classe 2	10 metros
Classe 3	1 metro

Fonte: *Bluetooth* SIG, b.

Uma das características desses produtos é a de consumir pouca energia, o que torna a tecnologia atrativa neste aspecto também. Por exemplo, sistemas usando a classe 2 consomem em torno de 2,5 mW e são os mais usados (*Bluetooth* SIG, b).

2.1 Arquitetura *Bluetooth*

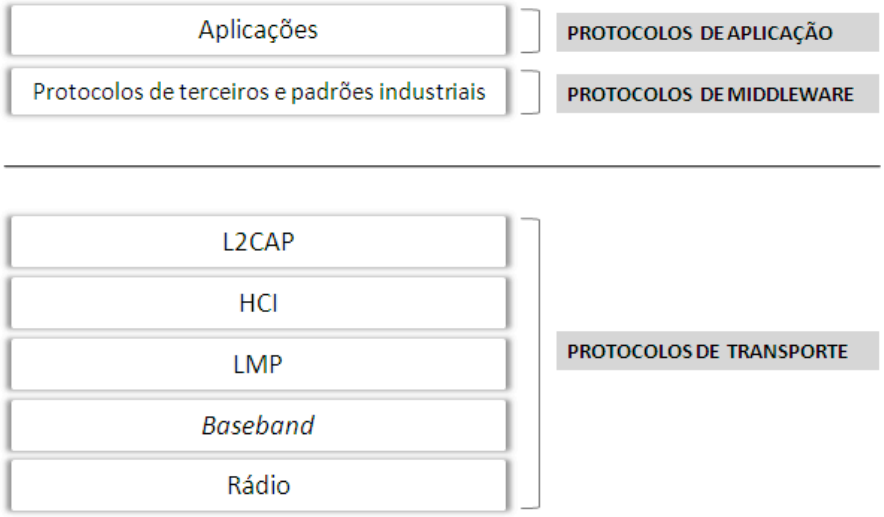
Os protocolos de comunicação, necessários para que esta tecnologia seja implementada, são divididos em três categorias:

- protocolos de transporte;
- protocolos de *middleware*;
- protocolos de aplicação.

Os protocolos de transporte tem por finalidade implementar e gerenciar as conexões física e lógica entre os dispositivos conectados à rede. Assim, controla a taxa de transmissão, define o canal de comunicação, envia e recebe os pacotes de dados e visa garantir estabilidade na interação entre componentes da rede. Os protocolos de *middleware* tem o objetivo de tratar os dados oriundos dos protocolos de transporte para que estes possam ser utilizados pelos protocolos de aplicação. Além disso também recebe os dados vindos dos aplicativos e prepara-os para serem utilizados para transporte. Logo, os protocolos de aplicação implementam os aplicativos propriamente ditos que utilizarão o *Bluetooth* como meio de comunicação (Silva, 2009).

A Figura 1 demonstra a hierarquia descrita no parágrafo acima.

Figura 1 – Hierarquia de protocolos *Bluetooth*



Fonte: Silva, 2009.

2.1.1 Protocolo de Rádio

A camada de rádio é responsável pela implementação física já descrita na Seção 2. Portanto, especifica características como frequência de operação, potência de transmissão e as técnicas

de modulação, responsáveis por possibilitar o melhor uso do espectro de frequência licenciado.

2.1.2 Protocolo de Baseband

Esta camada, que atua acima do rádio, tem como função implementar a conexão lógica entre os dispositivos da rede. Assim, é responsável por ditar as regras de acesso ao meio, identificar os dispositivos da rede, definir o canal de comunicação, detectar erros, escolher o formato do pacote a ser enviado na transmissão, dentre outras funções (Silva, 2009).

2.1.3 Protocolo de LMP

O Protocolo de gerenciamento de enlace (LMP), como o próprio nome já diz, é responsável por gerenciar o enlace de comunicação entre dois dispositivos da rede. Assim, tem a função de definir as configurações que devem existir em ambos usuários para que a conexão seja estabelecida. O gerenciador de enlace tem como principais funções gerir a qualidade do serviço prestado (QoS – *Quality of Service*) e garantir comunicação segura (Silva, 2009).

Este protocolo utiliza mensagens específicas para realizar o gerenciamento do enlace. O gerenciador de conexão recebe estas informações, as analisa e então controla as camadas inferiores. Dentre essas mensagens, pode-se destacar:

- controle de conexão;
- segurança;
- informações de modo de operação;
- informações das requisições;
- modo de teste.

2.1.4 Protocolo de HCI

A interface de controle de *host* (HCI) gerencia os protocolos LMP e *baseband*, definindo suas regras de funcionamento. Assim, é o protocolo que acessa as funções *baseband*. Na definição desse protocolo, é possível dividi-lo em três segmentos:

- *driver*;
- *firmware*;
- transporte.

O segmento *driver* é incorporado pelo software do *Bluetooth*. Ou seja, faz parte da inteligência do controlador. Recebe notificações dos eventos ocorridos, trata a informação e encaminha comandos de resposta (Silva, 2009).

O segmento de *firmware* é incorporado pelo *hardware* do *Bluetooth*. Ou seja, é responsável pela utilização dos recursos físicos presentes no dispositivo. Assim, acessa as funcionalidades *baseband* e LMP, além de gerenciar o acesso aos registradores de status, controle e eventos (Silva, 2009).

O segmento de transporte provê a comunicação entre os dois segmentos anteriores. Encaminha informações sem utilizá-las. Portanto, atua entre o *driver* e o *firmware* (Silva, 2009).

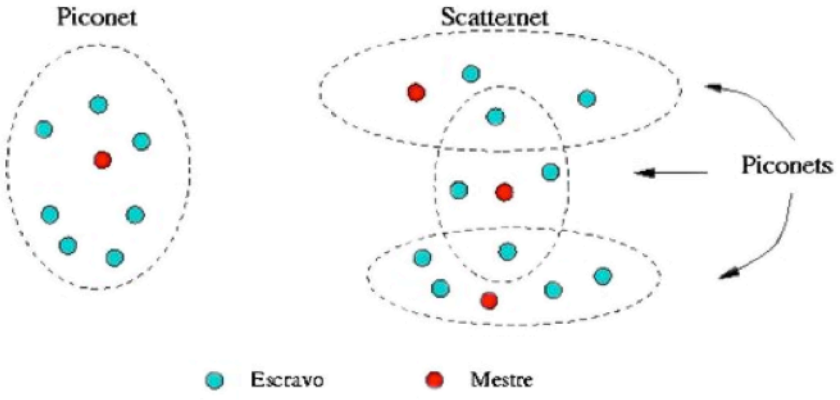
2.1.5 Protocolo de L2CAP

O protocolo de adaptação e controle do enlace lógico (L2CAP) é responsável por realizar a comunicação entre os protocolos de camada superior e os protocolos de camada inferior (Siqueira, 2006). Assim, o protocolo superior não precisa saber como o inferior funciona, simplesmente fornece seus dados para o L2CAP que é responsável por intermediar esta comunicação. Além disso, é responsável pela fragmentação e desfragmentação de pacotes.

2.2 Topologia da rede *Bluetooth*

A tecnologia *Bluetooth* introduz o conceito de *piconets*, que é simplesmente uma rede composta de até oito dispositivos que se comunicam utilizando este padrão. Dentro desta rede, um dispositivo é denominado mestre, ou dispositivo primário, que é o gestor da comunicação, responsável por definir o *clock* regente na rede e também a sequência de saltos em frequência padrão, e os outro denominados escravos ou dispositivos secundários. Uma *piconet* pode interagir com outra através de um de seus elementos, formando assim, uma *scatternet*. Diante dessa possibilidade, o alcance de comunicação pode ser estendido, ou seja, dispositivos não conectados diretamente podem se comunicar através dessa ponte criada (Siqueira, 2006). A Figura 2 demonstra esta configuração descrita.

Figura 2 – Redes *piconet* e *scatternet*



Fonte: Duarte; Langsh; Palomo; Xavier.

2.3 Estados de um dispositivo *Bluetooth*

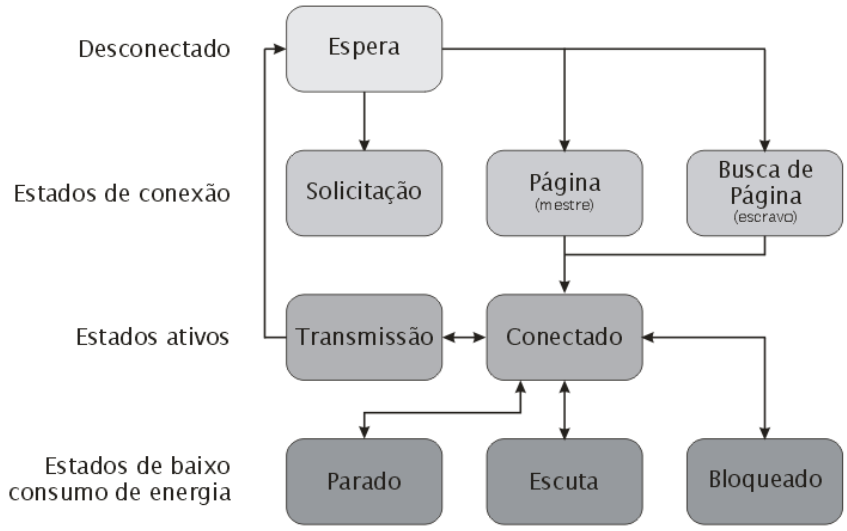
De acordo com Siqueira (2006), cada dispositivo *Bluetooth* pode estar em um dos seguintes estados:

- **Espera** – dispositivo está ativo, porém não está conectado a nenhuma *piconet*;
- **Solicitação** – quando é enviada uma solicitação de busca para encontrar outros dispositivos com quem possa se conectar;
- **Página** – um mestre de uma *piconet* já existente fica em estado de página quando envia mensagens visando encontrar novos dispositivos para se juntar à rede;

- **Conectado** – quando uma comunicação é bem sucedida entre um mestre e um novo dispositivo, este passa a ser escravo deste mestre e então recebe um endereço que o identifica na *piconet*;
- **Transmissão** – estado em que se encontra um escravo no momento em que transmite dados. Após finalizada a transmissão, volta ao estado de conectado;
- **Bloqueado** – escravo ativo, porém não emite nem recebe dados. Neste estado é verificada uma economia de energia;
- **Escuta** – um escravo fica em estado de escuta com o objetivo de economizar energia. Ou seja, quando não é a vez deste escravo transmitir informação ele desliga algumas funções que não serão utilizadas naquele momento e volta a religá-las quando solicitado;
- **Estacionado** – quando um escravo não utiliza mais a rede, o mestre pode colocá-lo neste estado. Ou seja, o escravo perde seu endereço de *piconet*, que pode ser repassado a outro dispositivo que queira utilizar a rede.

A Figura 3 demonstra de maneira ilustrativa os estados definidos anteriormente.

Figura 3 – Diagrama de estados para dispositivos *Bluetooth*



Fonte: Siqueira, 2006.

3 SISTEMA OPERACIONAL ANDROID

Conforme já citado no Capítulo 1, o sistema operacional *Android* foi escolhido para ser a plataforma base para o desenvolvimento de um programa que servirá de enlace entre o operador e o dispositivo a ser acessado.

Este sistema operacional foi desenvolvido com base no sistema *Linux* já existente e consolidado no mercado. Por ser uma plataforma de desenvolvimento livre, o *Android*, rapidamente, se tornou popular no mercado de eletrônicos móveis (Querino Filho, 2013).

De acordo com Querino Filho (2013), outro fator que fez com que a disseminação dessa tecnologia fosse tão rápida foi o fato de utilizar a linguagem de programação *Java* para implementar seus programas, linguagem esta já muito popular e explorada. Esta popularidade pode ser comprovada observando dados oficiais da própria Oracle (Empresa detentora da tecnologia Java), em que se observa que 97% dos computadores corporativos executam o Java e 89% dos computadores nos Estados Unidos executam Java (Java)

Primeiramente, vale ressaltar que foram utilizadas duas interfaces de programação. A primeira delas é o ambiente de programação Eclipse, em que é possível criar um aplicativo *Android* utilizando linhas de código através da linguagem Java e o MIT App Inventor 2, que é um serviço web criado pelo Google em parceria com MIT (Instituto de Tecnologia de Massachusetts), em que é possível desenvolver um aplicativo *Android* através de uma interface gráfica. A segunda interface de desenvolvimento foi utilizada devido à sua simplicidade de implementação e ampla utilização, como GAIDOS (2012). (MIT *App Inventor, a*). Porém, em uma etapa específica do projeto, que será detalhada na sequência deste trabalho, foi percebido que esta interface de programação não seria capaz de implementar uma funcionalidade necessária para o correto funcionamento do sistema. Portanto, foi preciso recorrer ao Eclipse, dado que esta ferramenta é capaz de implementar tal funcionalidade devido à sua versatilidade de programação.

A seguir serão apresentadas as duas interfaces de programação utilizadas neste trabalho.

3.1 Interface de programação Eclipse

3.1.1 Ferramentas necessárias

O primeiro passo para se programar aplicativos para *Android* utilizando o Eclipse é obter as ferramentas de implementação necessárias para isso, que são:

- ***Java* JDK7**: o *Java Development Kit,* ou seja, *kit* de desenvolvimento *Java* contém as ferramentas e bibliotecas essenciais para se implementar os programas;
- ***Android* SDK**: o *Android Software Development Kit*, ou seja, *kit* de desenvolvimento de software traz outros dois elementos essenciais para que a implementação seja feita com êxito.
- As bibliotecas *Java* que são de uso específico para a plataforma *Android*, dado que estas já estão incluídas no Eclipse e no *Android* SDK, por padrão.

3.1.2 Apresentação do Eclipse

O Eclipse é a interface que o programador utiliza para desenvolver e simular o código implementado. Este software pode ser utilizado para implementar algumas linguagens, porém para desenvolver aplicativos *Android* deve ser utilizada a interface padrão (Querino Filho, 2013).

A tela de apresentação do programa apresenta os seguintes campos:

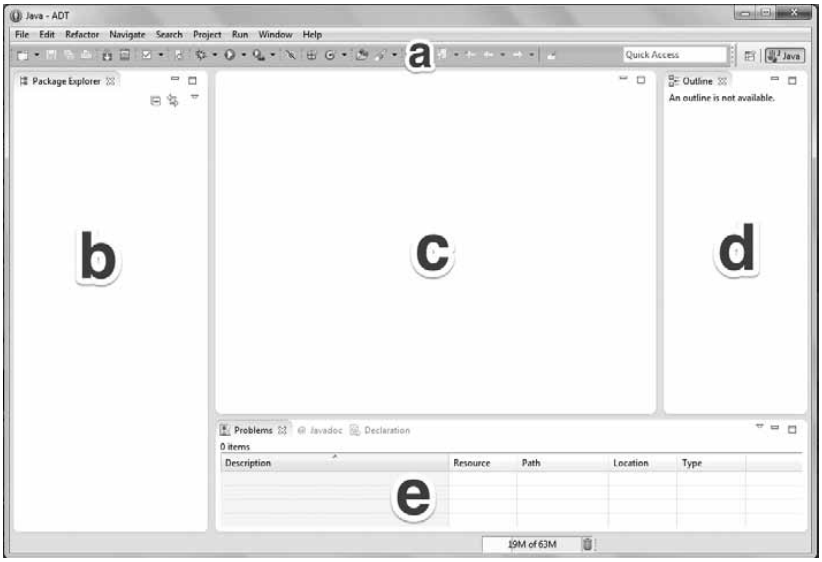
a) barra de ferramentas;
b) *package explorer*: explorador de pacotes, área de acesso aos projetos existentes;
c) editor de código: janela onde o código deve ser escrito;
d) *outline*: exibe elementos fundamentais do código aberto no editor de código. Faz a função de um índice;
e) *problems*: problemas, janela que apresenta os problemas encontrados na compilação do código.

A Figura 4 mostra onde se localizam esses elementos.

Figura 4 – Tela de apresentação do Eclipse

a) Barra de ferramentas; b) Package Explorer; c)Editor de Código;

d) Outline; e) Problems



Fonte: Querino Filho, 2013.

3.1.3 Integração *Bluetooth* e *Android*

Como neste trabalho o aplicativo *Android* será desenvolvido com o intuito de se utilizar a capacidade *Bluetooth* do celular ou *tablet* onde será executado, aqui serão apresentadas as classes *Java* que serão necessárias para tal utilização.

Primeiramente, vale ressaltar que estas classes são internas ao pacote *android.bluetooth* da linguagem *Java*. As classes a serem usadas são, *BluetoothAdapter*, *BluetoothDevice* e *BluetoothSocket* (Ableson, 2012).

O Quadro 2 demonstra a função de cada classe.

Quadro 2 – Classes *Java* para acessar o *Bluetooth* do dispositivo *Android*

Classe	Comentário
BluetoothAdapter	Essa classe representa os construtores locais para o hardware e a interface *Bluetooth* do dispositivo *Android*. Tudo começa com a *BluetoothAdapter*.
BluetoothDevice	Qualquer dispositivo remoto é representado como um *BluetoothDevice*.
BluetoothSocket	O *BluetoothSocket* é usado para troca de dados. Em uma nota mais prática, um dispositivo primário inicia uma conexão de soquete com um dispositivo secundário criando primeiro um *BluetoothSocket*.

Fonte: Ableson, 2012.

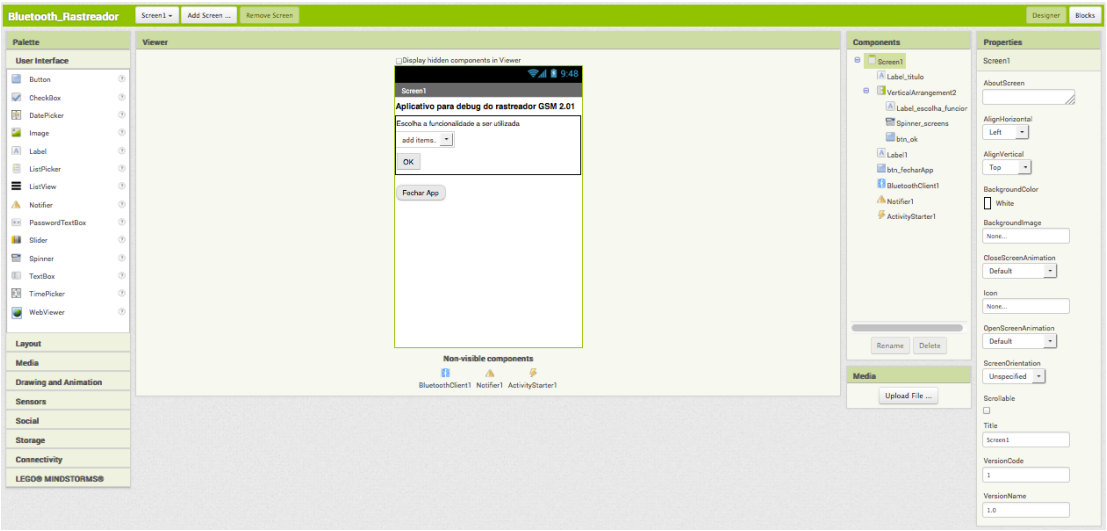
Com essas classes apresentadas anteriormente, é possível realizar as principais funcionalidades desejadas, como emparelhar dois dispositivos, ou seja, criar uma conexão segura entre eles (isso quer dizer criptografada, pois o *Android* só se conecta a outro dispositivo dessa maneira), monitorar esta conexão e trocar dados (Ableson, 2012).

3.2 MIT *App Inventor* 2

Desenvolvido inicialmente pelo professor Hal Ableson do MIT, em parceria com a divisão de educação do Google, este aplicativo, administrado pelo centro de ensino de dispositivos móveis do MIT, funciona como um serviço web, ou seja, que possa ser acessado pela Internet. Hoje, este serviço é utilizado por cerca de dois milhões de usuários, em 195 países e tendo produzido aproximadamente quatro milhões e meio de aplicativos (MIT *App Inventor, a*).

Este ambiente de programação é composto por dois módulos: *Component Designer* e o *Blocks Editor*. O primeiro módulo permite ao programador construir a interface visual do aplicativo, inserir botões, caixas de texto, textos, componentes para utilizar funções de relógio, *Bluetooth*, serviços telefônicos, dentre outras funcionalidades. O módulo apresenta uma simulação do resultado final, criada de maneira interativa pelo programador, em uma janela similar a uma tela de *smartphone*. A Figura 5 apresenta este módulo (Gomes, 2012).

Figura 5 – Módulo *Component Designer* do MIT *App Inventor*

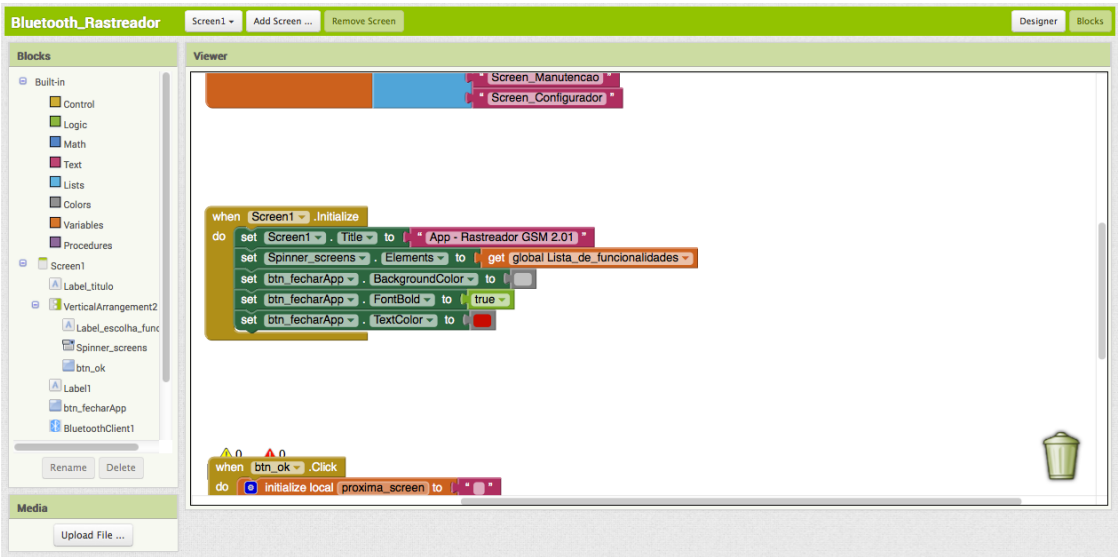


Fonte: Produção do próprio autor.

Na Figura 5, é possível observar as subdivisões de tela. À esquerda se encontram os possíveis componentes para a interface, como botões, caixas de texto, dentre outros. Ocupando a maior parte de tela se observa a interface gráfica que foi desenvolvida e que aparece no aplicativo pronto. Logo à direita aparece uma estrutura de blocos que exemplifica a estrutura montada na tela e no canto direito é possível configurar algumas características do componente selecionado, como por exemplo a cor do texto de um botão.

O segundo módulo, o *Blocks Editor* tem a funcionalidade de realizar a programação dos componentes inseridos à tela no módulo anterior. Esta ferramenta utiliza uma estrutura gráfica para realizar esta programação. São utilizados blocos que são interligados como peças de um quebra-cabeça e realizam as funcionalidades dos componentes já citados. Estes blocos representam ações a serem tomadas pelo sistema do aparelho, seja ele um *smartphone* ou um *tablet*, tratando dados e realizando as operações desejadas pelo programador, como por exemplo realizar uma conexão *Bluetooth*. A Figura 6 representa este módulo (Gomes, 2012).

Figura 6 - Módulo *Blocks Editor* do MIT *App Inventor*



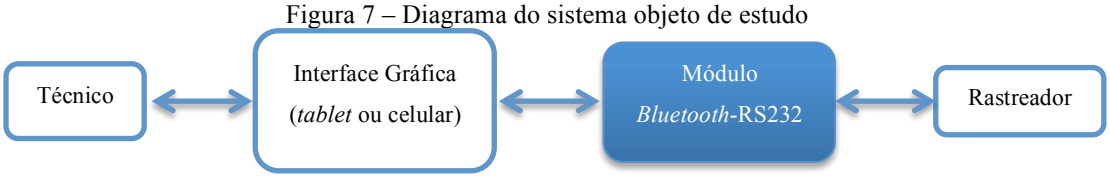
Fonte: Produção do próprio autor.

Na Figura 6 pode-se verificar como funciona este módulo. À esquerda é possível observar um campo onde estão os componentes que podem ser inseridos às funcionalidades da tela, como blocos de lógica, matemática, texto, lista, além das funcionalidades dos componentes já selecionados no outro módulo, como os botões e etc. Na tela central é onde são inseridos esses blocos e montada a estrutura lógica de programação.

4 PROJETO DO PROTÓTIPO BLUETOOTH – RS232

A Figura 7 apresenta um diagrama contendo as partes integrantes do sistema objeto de estudo. Neste Capítulo será tratado a respeito do bloco em destaque neste diagrama.

Figura 7 – Diagrama do sistema objeto de estudo



Fonte: Produção do próprio autor.

4.1 Módulo HC-05

Com o objetivo de substituir cabos e fornecer mobilidade, foi desenvolvido um protótipo a ser exemplificado a seguir.

Utilizando o software de projeto de circuitos eletrônicos Eagle e partindo do princípio que seria utilizado o componente HC-05 (módulo *Bluetooth*-UART) foi desenvolvido um protótipo de um circuito para atender aos anseios já explicitados anteriormente no trabalho.

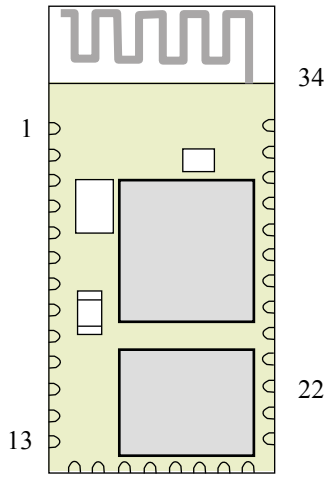
Figura 8 – Módulo *Bluetooth*-UART HC-05

Fonte: MSS Eletrônica

Este módulo *Bluetooth* foi escolhido devido à sua já consolidada utilização em diversos trabalhos. Ferreira (2013) é um caso exemplo deste uso.

O material de referência técnica deste módulo descreve a seguinte pinagem:

Figura 9 – Pinagem do módulo HC-05



Fonte: Itead Studio, 2010.

Quadro 3 – Pinagem detalhada do módulo HC-05

PINO	NOME	DESCRIÇÃO
1	UART TX	Transmissor para comunicação serial
2	UART RX	Receptor para comunicação serial
3	UART CTS	Controle de fluxo para comunicação serial
4	UART RTS	Controle de fluxo para comunicação serial
11	RESET	Entrada para resetar o dispositivo
12	VCC	Entrada de Tensão (+3,1Vdc até +4,2Vdc)
13	GND	Referencial Terra
15	USB D-	Barramento Dados – para USB
20	USB D+	Barramento Dados + para USB
21	GND	Referencial Terra
22	GND	Referencial Terra
31	LED 1	Saída para LED 1
32	LED 2	Saída para LED 1
34	KEY	Chave seletora de função configuração

Fonte: Itead Studio, 2010.

4.1.1 Configuração do módulo HC-05

Conforme descrito no Quadro 3, o pino 34 do módulo HC-05 tem a funcionalidade de selecionar o modo de configuração deste componente. Neste modo é possível se alterar e verificar alguns campos de informação acerca do funcionamento deste módulo (Itead Studio, 2010).

Para realizar esta configuração, é necessário conectar este módulo a um computador que possua uma interface de comunicação simulando uma comunicação terminal. Para este fim, foi utilizado o software HyperTerminal, nativo do Windows (HILGRAEVE).

O módulo HC-05 vem configurado para que sua comunicação aconteça a uma taxa de 38400bps, utilizando 8 bits por palavra e 1 bit de parada. Logo, é preciso configurar a conexão terminal com estas características. Para que o módulo permita acesso a estas configurações, é necessário, antes de energizar a placa, conectar o seu pino 34 à tensão de 3.3V. Após ligar o circuito, os comandos descritos no Quadro 4 podem ser utilizados para os fins de configuração. Existem outros comandos disponíveis que podem ser encontrados no manual de referência técnica (Itead Studio, 2010).

Quadro 4 – Comandos de configuração do módulo HC-05

Comando	Resposta	Parâmetros	Funcionalidade do comando
AT	Ok		Teste
AT+RESET	Ok		Realizar um reset via software
AT+ORGL	Ok		Restaurar configurações de fábrica
AT+ADDR?	+ADDR:<param> Ok	Endereço do hardware *Bluetooth*	Obter endereço do hardware *Bluetooth*
AT+NAME?	+NAME:<param> Ok	Nome do dispositivo *Bluetooth*	Obter o nome do dispositivo *Bluetooth*
AT+NAME=<param>	Ok		Alterar o nome do dispositivo
AT+ROLE?	+ROLE:<param> Ok	Função a ser desempenhada: 0-Escravo; 1-Mestre; 2-Escravo em loop	Obter o modo de operação do módulo *Bluetooth*
AT+ROLE=<param>	Ok		Alterar o modo de operação do módulo *Bluetooth*
AT+PSWD?	+PSWD:<param> Ok	Código PIN (Usado para pareamento *Bluetooth*)	Obter o PIN
AT+PSWD=<param>	Ok		Alterar o PIN
AT+UART?	+UART:<param1>, <param2>,<parm3> Ok	Param1: Taxa de transmissão da comunicação Param2: Número de bits de parada Param3: Paridade	Obter parâmetros de configuração da comunicação UART
AT+UART=<param1>, <param2>,<parm3>	Ok		Alterar parâmetros de configuração da comunicação UART

Fonte: Itead Studio, 2010.

Para realizar a comunicação serial com o equipamento rastreador, foi utilizada a seguinte configuração UART:

- Taxa de transmissão: 9600bps;
- Bits de parada: 1
- Paridade: 0

Além disso, outra configuração feita foi de colocar o sistema *Bluetooth* para operar no modo escravo.

4.2 Projeto do hardware

Com a configuração do hardware deste módulo em mãos foi iniciada a elaboração do projeto eletrônico. Utilizando o software Eagle, foi desenvolvido um projeto de um protótipo onde seria possível ter acesso a todas as funcionalidades disponíveis, mesmo as não utilizadas para este projeto. Isso, prevendo possíveis novas funcionalidades para este protótipo.

O Quadro 5 apresenta uma lista dos componentes eletrônicos utilizados no projeto. Sendo que, exceto as barras de pinos e o botão, estes são SMD (*Surface Mount Device)*.

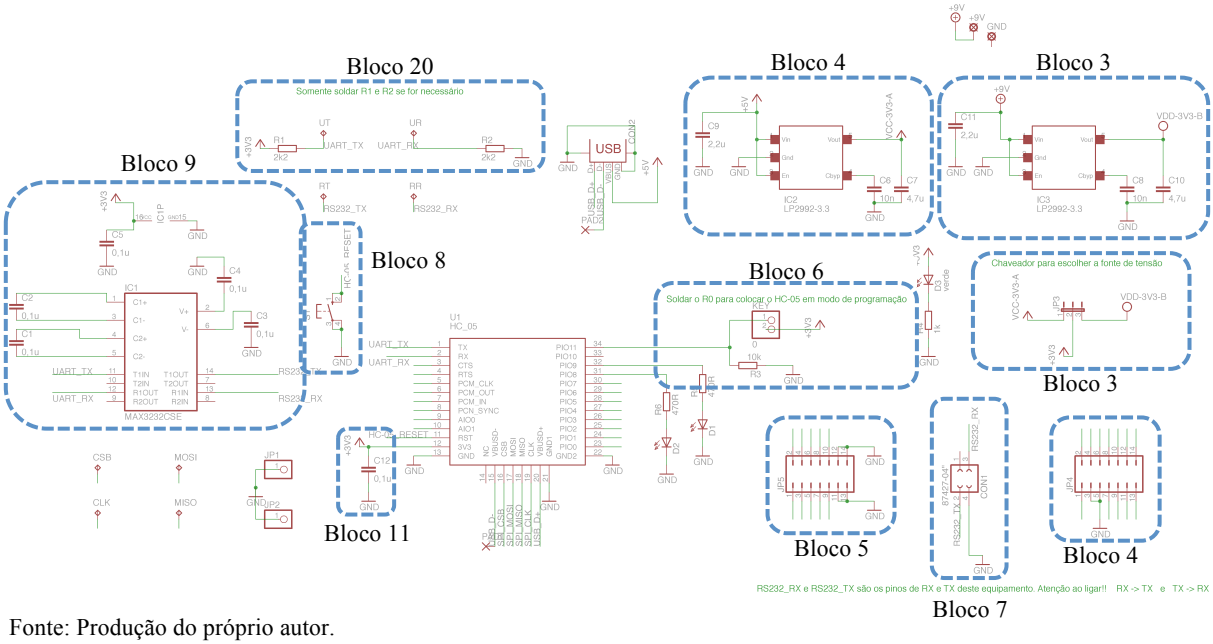
Quadro 5 - Lista de componentes do hardware projetado

Componente	Quantidade
Módulo HC-05	1
Regulador de tensão (LP2992-3.3V)	2
MAX3232CSE	1
Conector USB Macho	1
Conector Mini-Fit (4 posições)	1
Barra de pinos (14 posições)	2
Barra de pinos (3 posições)	1
Barra de pinos (2 posições)	1
Botão	1
Resistor (470 Ω)	2
Resistor (1k Ω)	1
Resistor (10k Ω)	1
Resistor (2,2k Ω)	2
Capacitor (0,1 μF)	6
Capacitor (2,2 μF)	2
Capacitor (4,7 μF)	2
Capacitor (10 nF)	2
LED	3

Fonte: Produção do próprio autor.

A Figura 10 mostra o esquemático do projeto desenvolvido.

Figura 10 – Esquemático do circuito eletrônico projetado



Fonte: Produção do próprio autor.

Vale ressaltar algumas observações acerca do projeto:

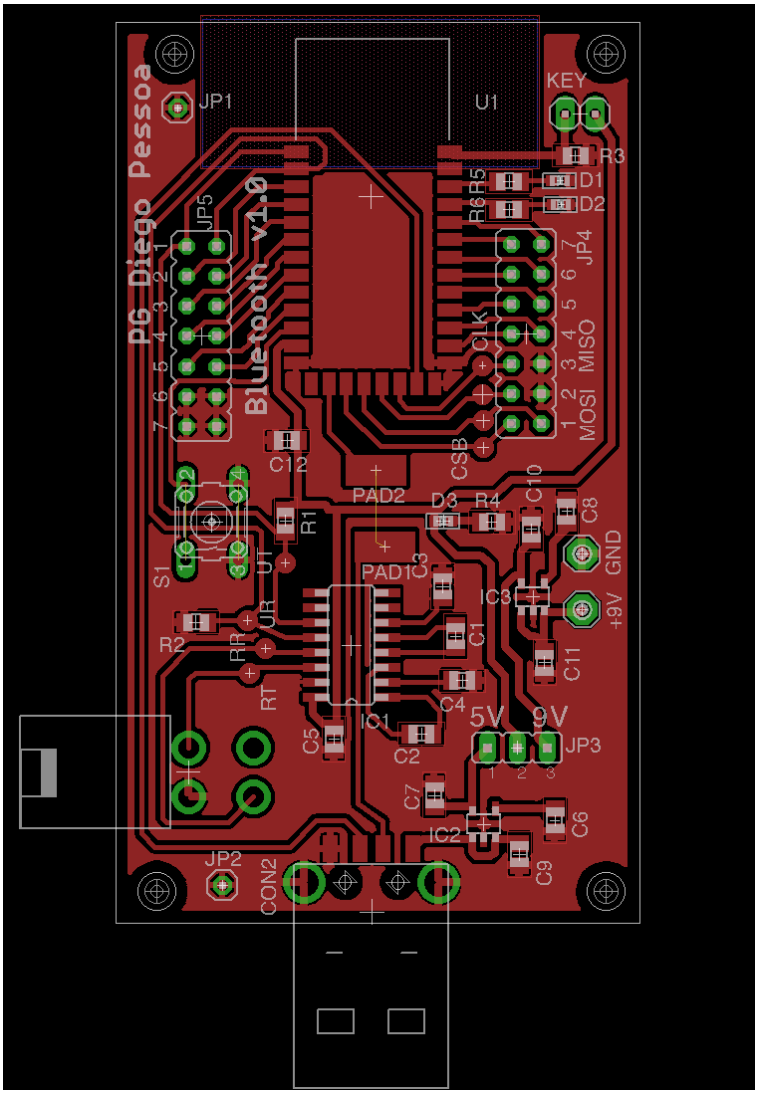
- foram projetadas duas fontes de alimentação possíveis para o circuito, a barra de pino JP3 (Bloco 3) permite que uma delas seja escolhida. A primeira é utilizando a tensão de 5V vinda de uma eventual conexão USB e a segunda maneira é através de uma fonte externa, projetada para ser de 9V (com o intuito de se utilizar uma bateria de 9V) porém pode ser utilizada uma fonte de tensão fornecendo de 3,3V até 16V. Sendo todas tensões contínuas. Estes circuitos alimentadores podem ser observados na Figura 10, destacados pelos Blocos 1 e 2. Utilizando os reguladores de tensão LP2992-3.3V foram projetados de acordo com o documento de referência (*Texas Instruments,* 2013);
- para fornecer acesso à todos os pinos do módulo HC-05, foram inseridas as barras de pino JP4 (Bloco 4) e JP5 (Bloco 5). Porém, não serão utilizadas neste projeto, é uma previsão futura;
- dado que o módulo *Bluetooth*- UART pode operar em dois modos, a barra de pinos KEY (Bloco 6) serve para escolher em qual funcionalidade o módulo será operado. Caso seja feito um curto circuito entre os dois pinos o módulo poderá ser ligado na

função de configuração. Ou seja, será possível alterar algumas configurações do módulo HC-05 utilizando comandos enviados via comunicação serial. Caso não exista esse curto circuito, o módulo fica livre para receber conexão *Bluetooth* e realizar a comunicação desejada (Itead Studio, 2010);

- para fornecer os sinais de comunicação ao meio externo à placa, foi inserido na projeto um conector do tipo Mini-Fit (Bloco 7);
- para realizar um reset via hardware é necessário apertar o botão de reset. Um curto é realizado do pino 11 do HC-05 para o referencial terra. Ao soltar o botão, o curto é desfeito e o dispositivo é reiniciado (Bloco 8);
- para viabilizar a comunicação entre o HC-05 e o equipamento externo, ajustando os níveis analógicos do sinal serial de UART para RS-232 e vice versa, foi inserido o componente MAX3232 (Bloco 9). Vale ressaltar que, os capacitores de acoplamento utilizados seguiram o documento de referência do componente (*Texas Instruments,* 2014);
- com o objetivo de assegurar os níveis lógicos 1 e 0 na vias de comunicação UART, foram inseridos no projeto os resistores R1 e R2 (Bloco 10), como resistores de pull-up e pull-down, respectivamente. Porém, estão no projeto para serem utilizados caso seja necessário (Resistor Guide);
- com o intuito de estabilizar a tensão de alimentação do módulo HC-05, foi inserido no circuito o capacitor C12 (Bloco 11).

A Figura 11 demonstra o projeto da placa desenvolvido a partir do esquemático anterior.

Figura 11 – Projeto final do hardware.



Fonte: Produção do próprio autor.

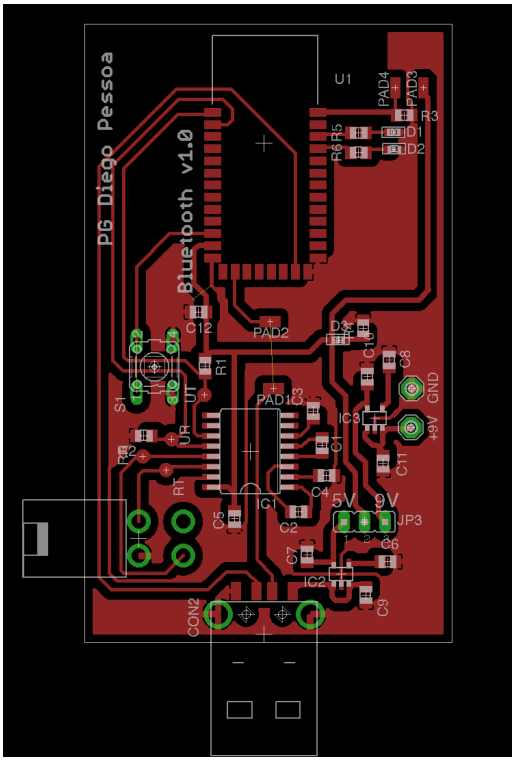
Acerca desta placa, vale ressaltar que não foi inserido plano de terra abaixo da antena *Bluetooth*, visando minimizar possíveis interferências (topo da figura). Além disso, para não ser necessário utilizar uma placa de camada dupla, deve existir uma conexão externa entre os PAD's 1 e 2 (centro da figura).

Considerando limitações de orçamento que foram encontradas para a confecção do protótipo, foi desenvolvido um projeto, adaptado desse apresentado anteriormente, que fosse mais fácil de confeccionar a placa de circuito impresso manualmente. Isso foi necessário devido ao pequeno espaçamento gerado naturalmente pelas barras de pinos JP4 e JP5 que foram

retiradas por não serem utilizadas para este projeto, sendo somente previstas com finalidades de utilização futura. Vale ressaltar que isto não afetou o funcionamento do circuito.

A Figura 12 apresenta este projeto simplificado.

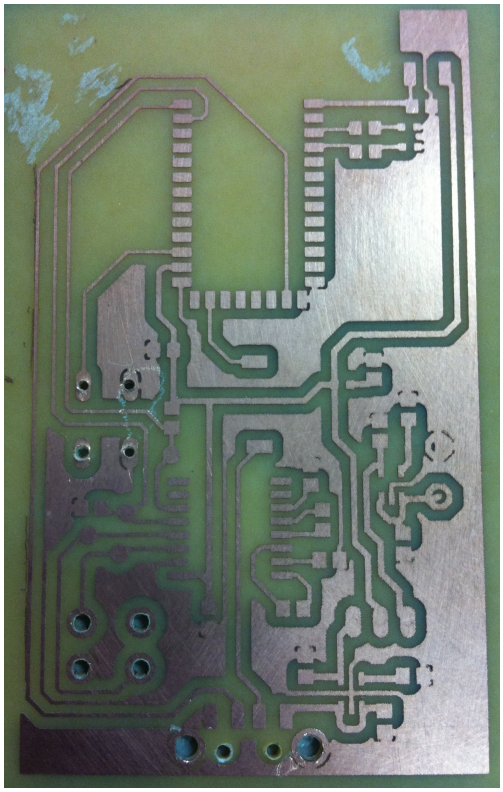
Figura 12 – Projeto simplificado



Fonte: Produção do próprio autor.

Utilizando um método de confecção de placa de circuito impresso similar a INAPE (2010), foi confeccionada uma placa de circuito impresso que pode ser observada na Figura 13, onde já se observa a placa antes da solda dos componentes.

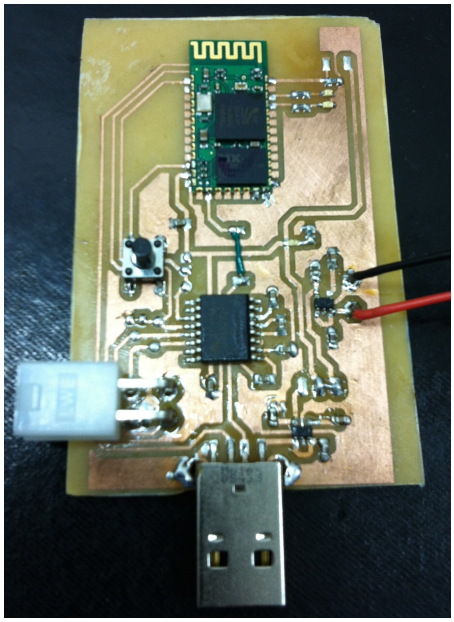
Figura 13 – Circuito impresso produzido



Fonte: Produção do próprio autor.

Após esta etapa de montagem, foram soldados os componentes eletrônicos e a placa passou a ter todas as condições para funcionamento pleno no seu uso. A Figura 14 exemplifica o protótipo pronto.

Figura 14 – Protótipo finalizado

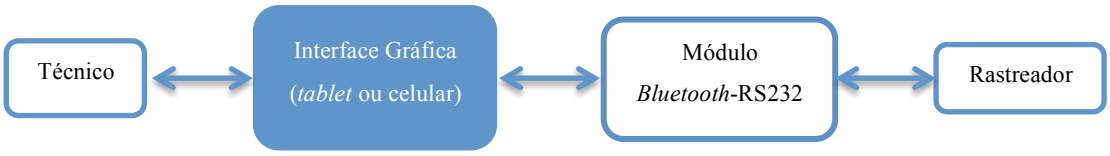


Fonte: Produção do próprio autor.

5 DESENVOLVIMENTO EM PLATAFORMA ANDROID

A Figura 15 apresenta, através do bloco em destaque, qual parte do sistema como um todo será tratada neste Capítulo.

Figura 15 - Diagrama do sistema objeto de estudo



Fonte: Produção do próprio autor.

5.1 Funcionalidades necessárias

Com o objetivo de configurar um equipamento específico, denominado rastreador, foi necessário, primeiramente, saber quais seriam as funcionalidades possíveis de serem configuradas e acessadas.

De acordo com documentação interna da Geocontrol Ltda, empresa desenvolvedora deste equipamento, foram obtidas essas informações. O Quadro 6 fornece uma lista de comandos, sendo definidos pelos seus respectivos propósitos, disponíveis para configurar e obter informações deste equipamento. Neste quadro, não serão apresentados os comandos propriamente ditos para não expor a segurança do equipamento que é comercializado pela empresa.

Quadro 6 – Lista de comandos disponíveis para configurar o rastreador

(Continua)

Categoria	Explicação do comando
Analisador	Debug do wismo, mostra na tela a troca de dados entre microcontrolador e wismo
	Debug da serial, mostra as informações de debug da serial
	Debug gps (rmc)
	Debug gps gga
	Debug gps gsv
Manutenção	Auto teste somente
	Auto teste + wdt
	Auto teste + apagar ic3
	Auto teste + apagar ic2
	Auto teste + iniciar gps

Quadro 6 – Lista de comandos disponíveis para configurar o rastreador

(Conclusão)

Categoria	Explicação do comando
Configurador	Alterar dados de operadora GPRS
	Altera o telefone pertencente ao módulo (sem o código da localidade)
	Altera o telefone 1 da base
	Altera o telefone 2 da base
	Altera somente o ip principal
	Altera somente o ip de backup
	Altera somente o ip de manutenção
	Altera os intervalos de transmissão
	Altera o ID do rastreador
	Obter ID
	Obter dados de configuração
	Obter download
	Obter dados de navegação
	ACK das entradas digitais
	Ativar saída 1
	Ativar saída 2
	Ativar saída 3
	Ativar saída 4
	Desativar saída 1
	Desativar saída 2
	Desativar saída 3
	Desativar saída 4

Fonte: Produção do próprio autor.

Assim, o aplicativo desenvolvido é capaz de enviar cada um desses comandos descritos anteriormente.

Analisando a estrutura do comando, conclui-se que este é composto por duas partes. A primeira é o comando propriamente dito e a segunda é um campo chamado CRC (*Cyclic Redundancy Check*). Este último é simplesmente um campo de verificação para que o receptor seja capaz de atestar que o dado recebido é realmente o dado enviado. Isso se faz necessário para diminuir a probabilidade de que um comando seja assimilado erroneamente. Erros de bits podem ser gerados por interferências, cabeamentos defeituosos, ruídos, dentre outras causas.

A partir de documentação própria da Geocontrol Ltda, foi obtido um script para se gerar esse código CRC tendo em vista o comando a ser enviado. Foi nesse ponto em que foi observado que a ferramenta MIT *App Inventor* 2 não seria capaz de implementar tal função. Isso porque esta interface não possui a funcionalidade necessária para realizar operações lógicas bit a bit, necessárias para o cálculo deste código de verificação. Daí a necessidade de se utilizar o Eclipse também. Assim, foi criada uma estrutura em que o aplicativo principal utiliza a funcionalidade do aplicativo para calcular o CRC, recebe o resultado e continua com seu funcionamento normalmente.

Uma vez já explicado o porquê do aplicativo e o que ele deve executar, o item a seguir apresentará o que foi implementado através do Eclipse e no item seguinte, o que foi desenvolvido utilizando a ferramenta principal, o MIT *App Inventor* 2.

5.2 Aplicativo desenvolvido utilizando a ferramenta Eclipse

Conforme já introduzido, este aplicativo foi desenvolvido com o objetivo de gerar o código CRC para que o comando possa ser enviado de forma completa e o rastreador execute o desejado.

O Algoritmo CRC trata o dado a ser enviada como um bloco único representando um polinômio. Por exemplo, o byte 10001001 representa o polinômio x^7+x^3+1.

O CRC é calculado por sucessivas divisões deste polinômio por um outro polinômio padronizado seguindo uma lógica específica. Por exemplo, considerando o divisor 1011, ou seja, x^3+x+1, o CRC resultante seria 101, em hexa, 0x5. A Figura 16 demonstra como este resultado foi obtido.

Figura 16 – Cálculo exemplo do CRC

```
1 0 0 0 1 0 0 1   0 0 0 Foram incluídos três zeros ao final da palavra para ser possível realizar o procedimento de divisão por completo
1 0 1 1
-----------------------
0 0 1 1 1 0 0 1   0 0 0 Resultado da divisão 1 (Note que este resultado pode ser obtido através da operação lógica XOR)
    1 0 1 1
-----------------------
0 0 0 1 0 1 0 1   0 0 0 Resultado da divisão 2 (Note que esta divisão só ocorreu onde o bit não é zero, pois a divisão seria nula nesse caso)
      1 0 1 1
-----------------------
0 0 0 0 0 0 1 1   0 0 0 Resultado da divisão 3
            1 0   1 1
-----------------------
0 0 0 0 0 0 0 1   1 1 0 Resultado da divisão 4
              1   0 1 1
-----------------------
0 0 0 0 0 0 0 0   1 0 1 Resultado Final (CRC - 3 bits)
```

Fonte: Produção do próprio autor.

Assim, o CRC é calculado de forma análoga à demonstrada anteriormente. Porém é utilizado um padrão de 16 bits e não três como exemplificado.

Este código é facilmente decifrado no receptor. Simplesmente é preciso realizar uma operação análoga a esta demonstrada na Figura 16 utilizando a sequência estendida (nesse caso com o os três bits de CRC incluídos, ou seja, 10001001 101) e o resultado final deve ser igual a zero.

Como este aplicativo só será usado para realizar este cálculo e retornar o resultado, não é necessário que ele tenha interface gráfica. Portanto, ele é basicamente composto pelos arquivos MainActivity.java e AndroidManifest.xml que estão descritos no Apêndice B – MainActivity.java e AndroidManifest.xml. O primeiro arquivo é onde o programa propriamente dito é definido utilizando a linguagem de programação Java e o segundo é um arquivo de definições utilizado pelo sistema operacional para conceder permissões de uso para o aplicativo. Existem outros arquivos que são gerados por padrão e automaticamente pelo Eclipse para que o aplicativo seja completo, porém não serão detalhados neste trabalho (*Android*).

5.3 Aplicativo desenvolvido utilizando ferramenta MIT *App Inventor* 2

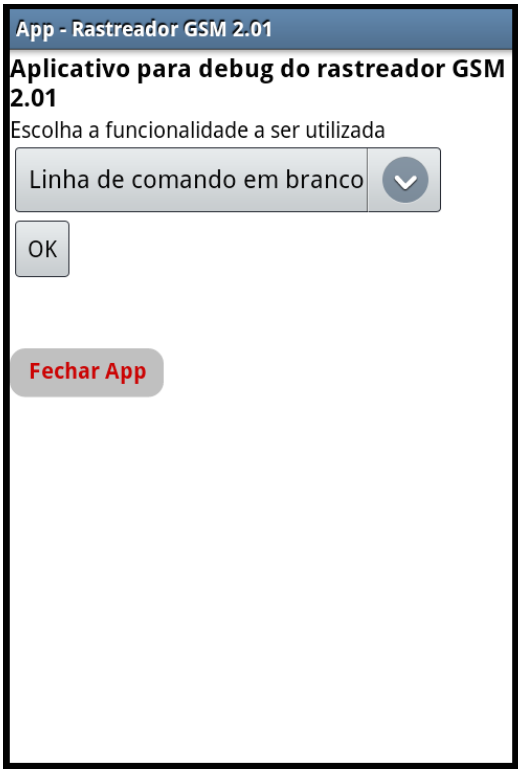
Este aplicativo foi desenvolvido tendo como objetividade ser capaz de executar os comandos listados no Quadro 6. Portanto, para fins de organização do aplicativo, ele foi criado de forma que existem cinco telas de visualização disponíveis:

a) tela Inicial;
b) tela para envio de comando não listado, ou seja, com campo em branco para ser preenchido pelo operador;
c) tela para envio de comandos da categoria Analisador;

d) tela para envio de comandos da categoria Manutenção;
e) tela para envio de comandos da categoria Configurador.

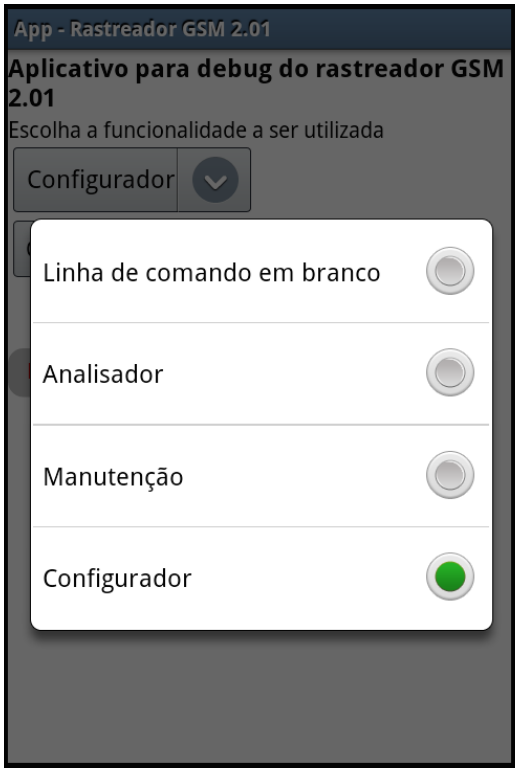
Da tela inicial é possível acessar qualquer uma das outras telas, selecionando qual a tela de destino desejada pela lista disponível. A Figura 17 demonstra a tela inicial do programa e a Figura 18 demonstra a lista expandida com as opções de tela disponíveis.

Figura 17 – Tela inicial do aplicativo



Fonte: Produção do próprio autor.

Figura 18 – Listagem das telas disponíveis



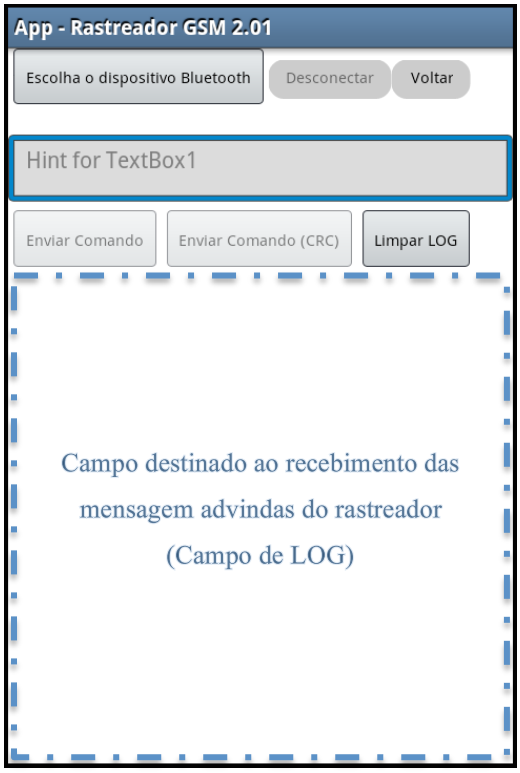
Fonte: Produção do próprio autor.

5.3.1 Tela com campo em branco

Nesta tela, o usuário do aplicativo tem a possibilidade de enviar um comando não listado no Quadro 6. Isso fornece uma versatilidade ao aplicativo no sentido de possibilitar que este aplicativo seja utilizado para acesso à algum outro equipamento que também utilize o mesmo padrão de comandos. Por exemplo, rastreadores mais antigos, também projetados pela Geocontrol, porém utilizam outros comandos.

Primeiramente, será apresentado o processo padrão para realizar a conexão entre o aplicativo e o módulo *Bluetooth*. Observando a Figura 19 que demonstra a tela em questão, é possível verificar o botão “Escolha o dispositivo Bluetooth” que ao ser selecionado fornece uma lista dos dispositivos disponíveis para realizar a conexão.

Figura 19 – Tela com campo em branco



Fonte: Produção do próprio autor.

Após escolhido o dispositivo, ao selecioná-lo na lista exibida pelo aplicativo, aparecerá uma mensagem abaixo do botão citado anteriormente confirmando que a conexão foi estabelecida com sucesso, como se observa na Figura 20.

Figura 20 – Conexão estabelecida

Fonte: Produção do próprio autor.

Após esta etapa, os botões “Desconectar”, para terminar a conexão, o “Enviar Comando”, para envio de comandos, e o “Enviar Comando (CRC)”, que calcula o CRC para formar o comando completo e então enviá-lo, são habilitados. Ao clicar no botão “Enviar Comando

(CRC)", este aplicativo executa uma sequência de ações para que o envio do comando seja bem sucedido. Dentre essas ações, este aplicativo executa uma chamada do aplicativo já apresentado no item 5.2 que tem a função de receber o comando digitado, calcula o CRC e o envia de volta para o aplicativo principal. Ou seja, neste caso o operador digitou um comando sem o campo CRC. Neste ponto, o comando é formado por completo, unindo as duas partes e é enviado. No caso do uso do botão "Enviar Comando", o operador precisa digitar o comando completo, por exemplo "comandoA0E21", em que "0E21"é o código CRC.

O campo abaixo dos botões "Enviar Comando", "Enviar Comando (CRC)" e "Limpar LOG", que pode ser verificado na Figura 19, é reservado para que as respostas do equipamento sejam apresentadas. E este terceiro botão citado tem a função de apagar este campo.

O botão "Voltar" volta para a tela inicial e termina a conexão caso ela ainda esteja ativa.

5.3.2 Telas para envio de comandos

Devido ao fato de que as telas para envio de comandos das categorias analisador, manutenção e configurador terem a interface gráfica e o funcionamento similar, serão aqui apresentadas figuras das três telas, porém só explicando o funcionamento de uma. A Figura 21 apresenta as três telas citadas. Sendo nesta ordem, tela do analisador, tela de manutenção e tela do configurador.

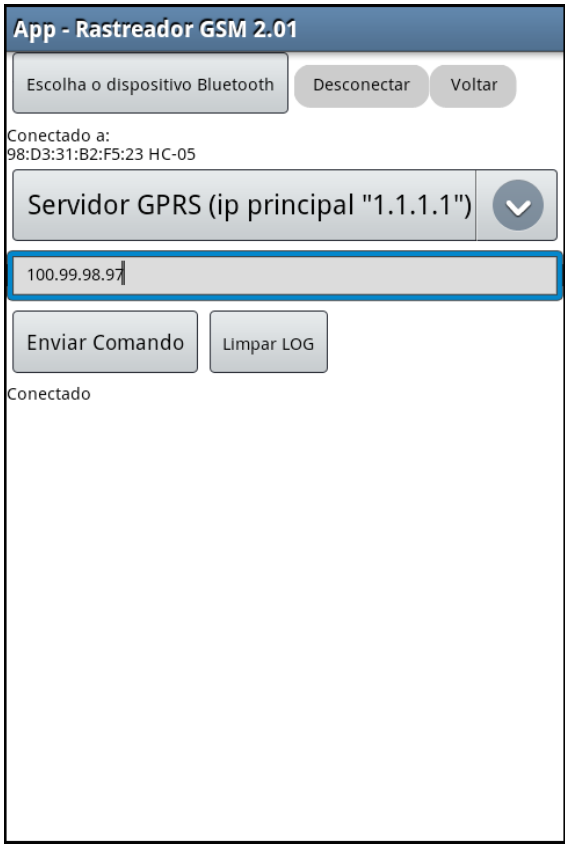
Figura 21 – Telas para envio de comandos

Fonte: Produção do próprio autor.

No item anterior, em que foi apresentada a tela com campo para livre preenchimento foi explicado a respeito de como realizar a conexão *Bluetooth* com o módulo e a respeito do funcionamento lógico de operação dos botões. Nestas telas, o funcionamento é análogo a este já descrito, porém não é possível enviar qualquer comando, somente os já disponíveis.

Uma diferença que pode ser observada é que na tela da categoria configurador existe um campo para digitação. Este campo só é habilitado quando necessário. Ou seja, caso o comando a ser enviado requisite algum parâmetro, assim este campo é liberado para preenchimento. Por exemplo, na Figura 22 é possível observar o comando "Servidor GPRS (ip principal "1.1.1.1")" em que o campo de digitação está habilitado. Assim, pode-se observar que ele foi preenchido com um endereço IP fictício somente para efeito de demonstração. Portanto, ao clicar no botão "Enviar Comando" o programa insere esta informação ao comando, chama o aplicativo para gerar o CRC, recebe a informação, monta o comando e o envia.

Figura 22 – Uso do campo de texto na tela da categoria configurador



Fonte: Produção do próprio autor.

6 CASO PRÁTICO – RASTREADOR

Este trabalho foi desenvolvido com o intuito de configurar um equipamento denominado rastreador. Portanto neste capítulo ele será apresentado e também será demonstrado um exemplo da comunicação bem sucedida entre o aplicativo e este dispositivo.

A Figura 23 apresenta o produto. Na Figura 24 se observa os conectores do tipo Mini-Fit pelos quais a comunicação e a alimentação são realizadas. Assim, é através deste conector que se deverá conectar o módulo *Bluetooth*-UART projetado no Capítulo 4.

Figura 23 - Rastreador

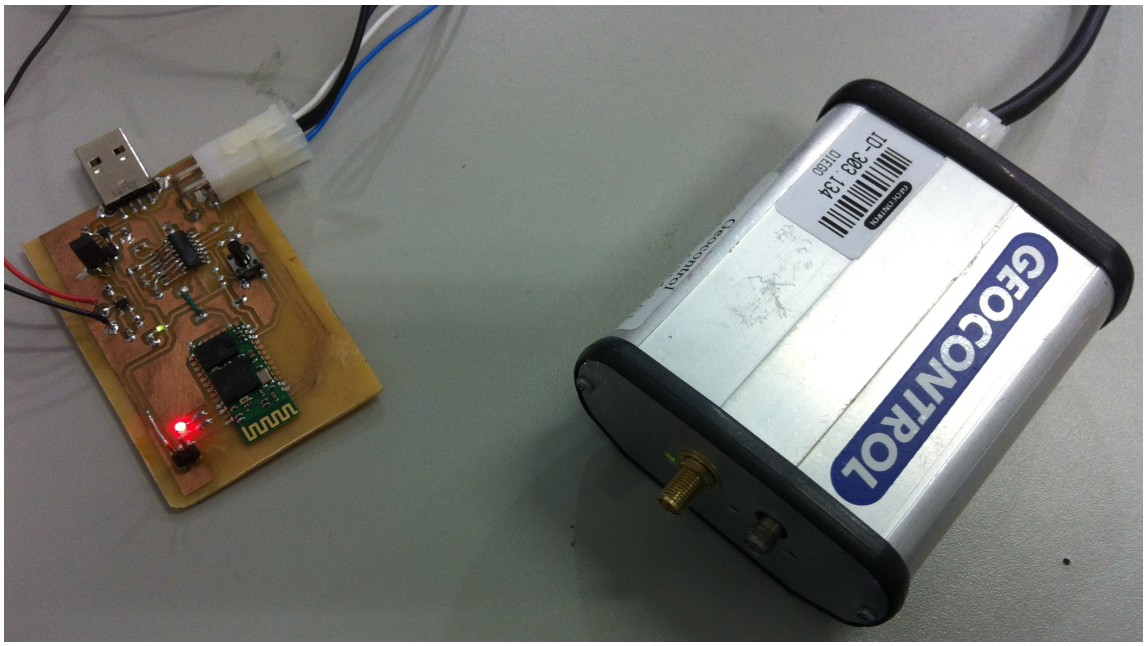
Fonte: Produção do próprio autor.

Figura 24 – Conectores Mini-Fit do rastreador

Fonte: Produção do próprio autor.

A Figura 25 demonstra como estes dois dispositivos foram conectados um ao outro. Desta forma, é simplesmente necessário que o aplicativo se conecte ao módulo pela conexão *Bluetooth* e envie os comandos desejados.

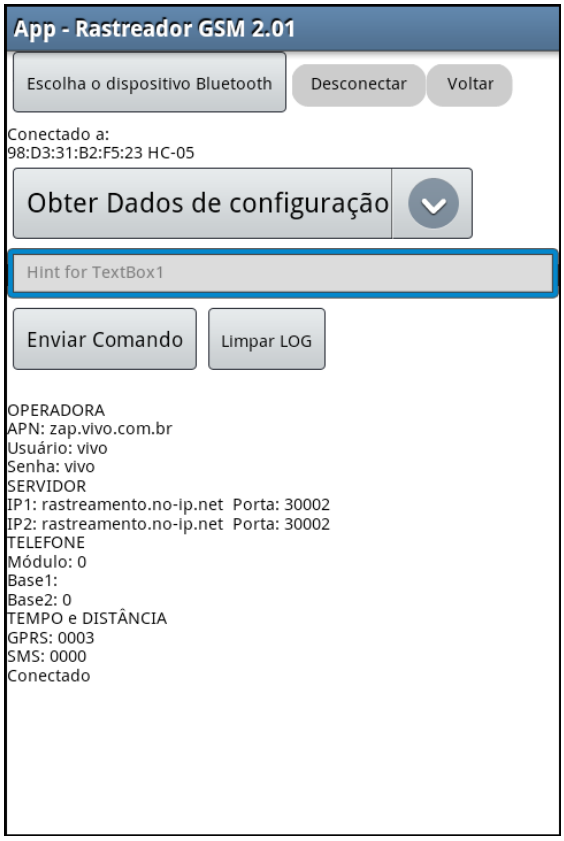
Figura 25 – Conexão entre o rastreador e o módulo *Bluetooth*-UART



Fonte: Produção do próprio autor.

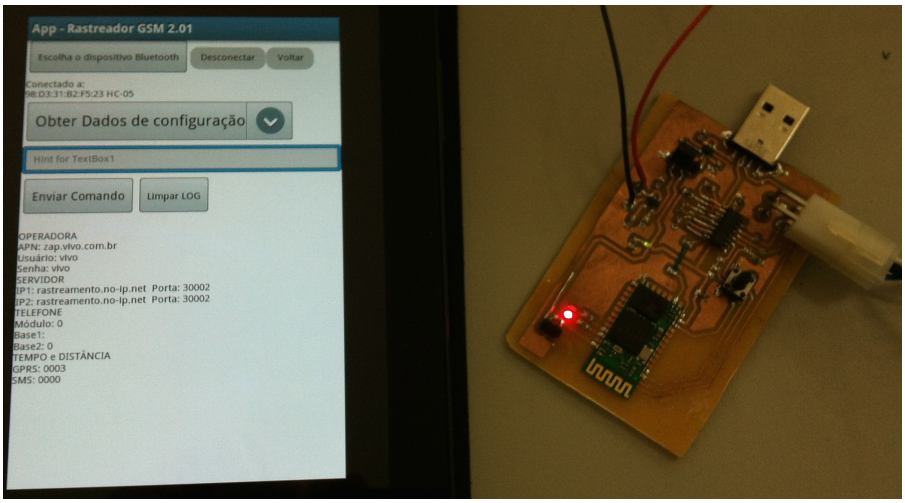
Com a finalidade de exemplificar o uso deste sistema, a Figura 26 demonstra a tela para envio de comandos da categoria configurador ao enviar o comando “Obter dados de configuração” e receber sua resposta. E a Figura 27 demonstra o aplicativo funcionando no *tablet* que foi utilizado para a realização deste trabalho ao lado do hardware projetado.

Figura 26 – Tela do configurador após receber resposta de comando exemplo



Fonte: Produção do próprio autor.

Figura 27 – Foto do aplicativo em operação



Fonte: Produção do próprio autor.

Como se pode observar na Figura 26, o campo destinado às respostas do rastreador fornece a resposta já tratada para este caso exemplo. Este comando requisita ao rastreador seus dados de configuração relacionados à comunicação GPRS. Este responde com estas informações

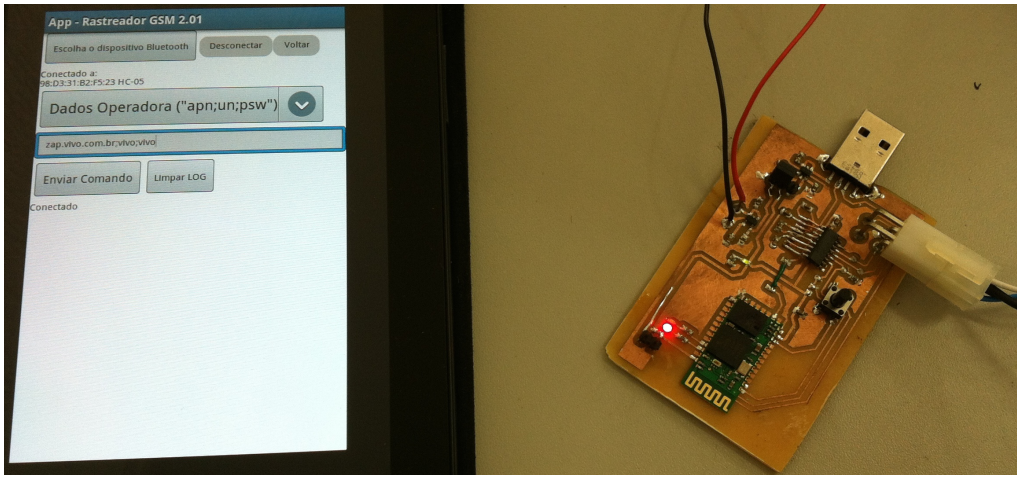
apresentadas. Vale ressaltar, que estes dados vão sofrer alterações em equipamentos diferentes.

A Figura 28 mostra a tela para envio do comando de configuração de características GPRS requisitadas ao rastreador no comando exemplo anterior. Assim é possível reconfigurar alguma informação que esteja incorreta. Na Figura 29 observa-se o aplicativo sendo executado no *tablet*.

Figura 28 – Tela com comando para configurar informações GPRS

Fonte: Produção do próprio autor.

Figura 29 – Foto do aplicativo na tela de envio do comando “Dados Operadora”



Fonte: Produção do próprio autor.

Um outro exemplo de comando que pode ser executado pelo técnico de manutenção é a requisição de *Debug* para o GPS, que pode ser de três tipos GGA (*Global Positioning System Fix Data*), GSV (*Satellites in view*) ou RMC (*Recommended Minimum sentence C*). Estes são comandos padronizados internacionalmente e fornecem informações específicas acerca do funcionamento deste sistema. Logo, este procedimento possibilita verificar se o rastreador está conectado devidamente ao sistema de rastreamento geográfico. A Figura 30 demonstra um exemplo desse caso. Vale ressaltar que o operador deve ser apto a realizar tal verificação tendo em vista que esse *Debug* simplesmente disponibiliza ao usuário do aplicativo as informações geradas pelo modem GPS sem tratamento algum.

Figura 30 – Tela Analisador ao executar o comando “Debug GPS GGA”

Fonte: Produção do próprio autor.

Assim, desta figura anterior é possível extrair que a última posição obtida por este rastreador foi nas coordenadas 20º17,3676’ S e 40º17,5212’L, aqui em Vitória mesmo, isso porque neste momento ele não estava conectado à rede de satélites. O que pode ser um problema que deve ser melhor explorado pelo técnico.

Além destas informações já apresentadas neste capítulo, foram realizadas medidas de consumo médio do hardware projetado em diferentes modos de operação e também verificada a distância em que a comunicação apresentou as primeiras falhas.

Os dados relativos ao consumo são apresentados no Quadro 7. Vale ressaltar que os valores de corrente apresentados são referentes a uma alimentação por uma fonte de tensão de 9V.

Quadro 7 – Consumo médio aproximado do hardware projetado

Operação	Consumo médio aproximado
Aguardando conexão *Bluetooth*	40 mA
Conectado porém sem transmissão de dados	15 mA
Conectado e com transmissão de dados	25 mA
Em estado de configuração via computador	15 mA

Fonte: Produção do próprio autor.

Além disso, foi verificado também que a comunicação sem fio já demonstrou as primeiras falhas quando se mantinha uma distância aproximada de 10 metros, com obstáculos, entre o *tablet* e o receptor *Bluetooth*. Dado que para uma média de 5 tentativas de comunicação 2 não configuravam o funcionamento correto do sistema.

7 CONCLUSÕES E SUGESTÃO DE TRABALHOS FUTUROS

Este trabalho se propôs a desenvolver uma interface de manutenção para acessar equipamentos instalados em campo utilizando a tecnologia de comunicação sem fio *Bluetooth*, visando substituir as técnicas cabeadas existentes. Para alcançar este objetivo foi necessário projetar e implementar um hardware para criar o enlace sem fio entre o equipamento a ser configurado e a interface gráfica de operação do técnico. Assim, foi desenvolvido também um aplicativo para funcionar em plataforma *Android* através de um celular ou *tablet*.

Para projetar o circuito eletrônico, foi utilizado o software Eagle, tendo como base as informações técnicas fornecidas pelos fabricantes dos componentes utilizados. Ao término do projeto, e implementação do mesmo, foi possível verificar que o circuito atendeu aos anseios iniciais do projeto, realizando a comunicação desejada.

Com a finalidade de desenvolver o aplicativo em questão foram utilizadas as ferramentas de programação Eclipse e MIT *App Inventor* 2, conforme já explicado no Capítulo 5. Após esta etapa de desenvolvimento concluída, foi possível atestar que o aplicativo é capaz de viabilizar a comunicação desejada e assim atender ao objetivo do trabalho.

A realização deste trabalho foi importante, pois criou dentro da empresa um novo procedimento mais simples e mais fácil para realizar a manutenção necessária em campo. Assim, fornecer acesso a rastreadores já instalados em veículos sem a necessidade de retirá-los de campo para se verificar qual o problema existente.

Sugere-se como trabalhos futuros, um estudo mais detalhado acerca de um outro componente eletrônico capaz de realizar comunicação *Bluetooth* que consuma menos energia e assim possa ser viabilizado um novo rastreador com esta tecnologia embarcada também. Este módulo HC-05 consume quase metade da energia de um rastreador em funcionamento, justificando a decisão de não incorporá-lo à este equipamento, portanto, este novo estudo pode ser interessante. Além disso, esta mesma interface *Bluetooth*-UART projetada pode proporcionar acesso a outros equipamentos embarcados. Sendo necessário simplesmente

configurá-lo de maneira adequada para ser compatível com o novo equipamento e desenvolver um novo aplicativo que já tenha os comandos pré-definidos.

8 REFERÊNCIAS BIBLIOGRÁFICAS

Ableson, W. F.; Sen, R.; King, C.; Ortiz, C. E. ***Android* em ação**. 3. ed. Rio de Janeiro: Elsevier Editora Ltda. 2012. cap. 14, p. 387-395.

Android. **App Manifest | Android Developers**. Disponível em: <http://developer.android.com/guide/topics/manifest/manifest-intro.html>. Acesso em: 27 nov. 2014.

Bluetooth SIG. **Fast Facts | *Bluetooth* Technology Website**. Disponível em: <http://www.bluetooth.com/Pages/Fast-Facts.aspx>. a. Acesso em: 26 jun. 2014.

Bluetooth SIG. **Basics | *Bluetooth* Technology Website**. Disponível em: <http://www.bluetooth.com/Pages/Basics.aspx>. b. Acesso em: 26 jun. 2014.

Cruz, J. C. S. S. **Automação residencial utilizando smartphone *Android*, *Bluetooth* e conversor RS-485**. 2013. Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) – Colegiado do Curso de Engenharia de Computação, Faculdade de Tecnologia e Ciências Sociais Aplicadas, Brasília, 2013.

Duarte, O. C. M. B.; Langsh, F. L.; Palomo, R. T.; Xavier, G. A. ***Bluetooth* – Camada física e camada de acesso ao meio**. Disponível em: <http://www.gta.ufrj.br/grad/09_1/versao-final/bluetooth/index.htm>. Acesso em: 29 jun. 2014.

Ferreira, E. P.; Alves, N. L. A. **Braço articulado com controle proporcional de movimento comandado via Bluetooth por um aplicativo desenvolvido para plataformas Android**. 2013. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica/Eletrônica) – Colegiado do Curso de Engenharia Elétrica/Eletrônica, Universidade do Vale do Paraíba, São José dos Campos, 2013.

GAIDOS, O. F. **Sistema Móvel de Monitoramento e Treinamento para Ciclista com Smartphone Android**. 2012. Dissertação de Mestrado em Engenharia Elétrica, Publicação PPGEA.DM 474/2012, Departamento de Engenharia Elétrica – Universidade de Brasília, Brasília, 2012.

Querino Filho, L. C. In: _____. **Desenvolvendo seu primeiro aplicativo *Android***. 1. ed. São Paulo: Novatec Editora Ltda. 2013. cap. 1, p. 15-70.

Gomes, T. C. S.; DE Mello, J. C. B. *App inventor for android*: uma proposta construcionista para experiências significativas de aprendizagem no ensino de programação. In: SIMPÓSIO HIPERTEXTO E TECNOLOGIAS NA EDUCAÇÃO, 4., 2012, Recife. **Anais Eletrônicos do quarto Simpósio Hipertexto e Tecnologias na Educação.** Recife: 2012.

HILGRAEVE. ***Hyperterminal Windows* 7 *Terminal Emulation Program* – HILGRAEVE**. Disponível em: <http://www.hilgraeve.com/hyperterminal/>. Acesso em: 26 nov. 2014.

INAPE. **Circuito impresso de qualidade com baixo custo | INAPE**. 2010. Disponível em: <http://www.inape.org.br/eletronica/circuito-impresso-de-qualidade-com-baixo-custo>. Acesso em: 26 nov. 2014.

Itead Studio. **DS_BluetoothHC-05.pdf**. 2010. Disponível em: <ftp://imall.iteadstudio.com/IM120417010_BT_Shield_v2.2/DS_BluetoothHC05.pdf>. Acesso em: 25 nov. 2014.

Java. **Obtenha informações sobre a Tecnologia Java**. Disponível em: <https://www.java.com/pt_BR/about/>. Acesso em: 4 dez. 2014.

MIT *App Inventor*. ***About us* | *Explore MIT App Inventor***. Disponível em: <http://appinventor.mit.edu/explore/about-us.html>. a. Acesso em: 17 nov. 2014.

MSS Eletrônica. **Módulo *Bluetooth* Serial RS232 TTL**. Disponível em: <http://www.msseletronica.com/loja/detalhes.php?urlid=821>. Acesso em: 17 nov. 2014.

Resistor Guide. **Pull up resistor / Pull down resistor**. Disponível em: <http://www.resistorguide.com/pull-up-resistor_pull-down-resistor/>. Acesso em: 25 nov. 2014.

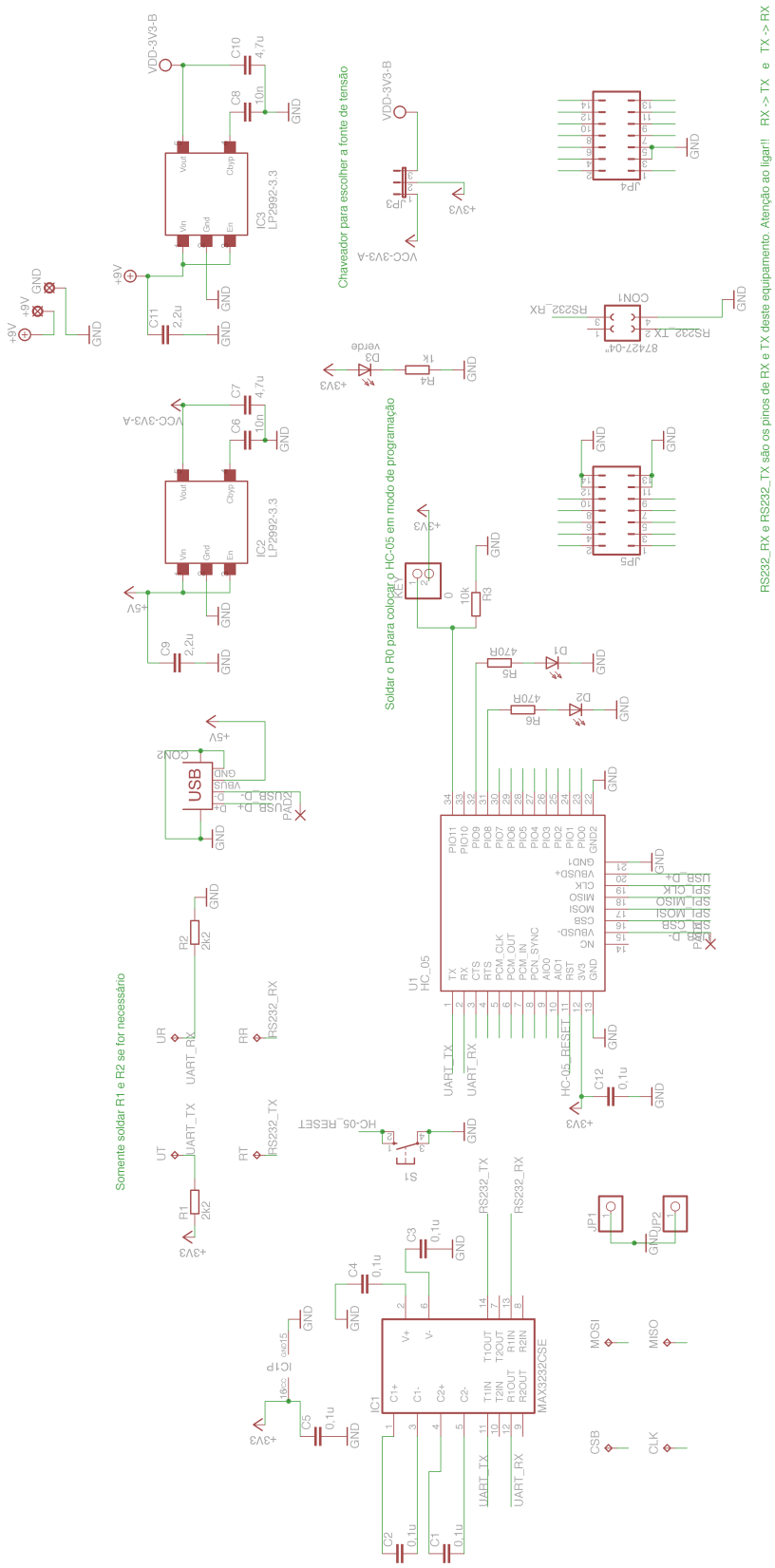
Silva, D. F. **Sistema de comunicação *bluetooth* utilizando microcontrolador**. 2009. Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) – Colegiado do Curso de Engenharia de Computação, Escola politécnica de Pernambuco – Universidade de Pernambuco, Recife, 2009.

Siqueira, T. S. ***Bluetooth* – Características, protocolos e funcionamento**. 2006. Disponível em: <http://www.ic.unicamp.br/~ducatte/mo401/1s2006/T2/057642-T.pdf>. Acesso em: 29 jun. 2014.

Texas Instruments. **LP2992**. 2013.

Texas Instruments. **MAX3232**. 2014.

APÊNDICE A – ESQUEMÁTICO DO CIRCUITO ELETRÔNICO PROJETADO



APÊNDICE B – MAINACTIVITY.JAVA E ANDROIDMANIFEST.XML

MainActivity.java:

```
package com.calcula_crc;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;

public class MainActivity extends Activity {

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

		// Obtem a Intent que inicializou esta aplicação
		Intent intent = getIntent();
		Uri dados = intent.getData();

		// Exexcuta cálculo do CRC tendo como parâmetro o comando de origem
		int crc_inicial = 0;
		String comando = dados.toString();
		String CRC = crc16_avlgsm201(comando, crc_inicial);

		// Gera Intent para retornar o resultado
		Intent result = new Intent("com.example.RESULT_ACTION", Uri.parse(CRC));

		setResult(Activity.RESULT_OK, result);
		finish();
	}

	public int crc16_byte(char val, int crc_inicial){
		// Função que gera o CRC para cada caracter do comando
		for (int li=0; li<8; li++){
			if (((val & 0x01) ^ (crc_inicial & 0x0001)) != 0) {
				crc_inicial = crc_inicial >> 1;
				crc_inicial = crc_inicial ^ 0xA001;
			}
			else {
				crc_inicial = crc_inicial >> 1;
			}
			val >>= 1;
		}
		return crc_inicial;
	}

	public int crc16_string(String val, int crc_inicial) {
		// Função que chama a função crc16_byte passando com parametros o caracter e o crc
acumulado
		for (int li = 0; li < val.length(); li++){
```

```
                        crc_inicial = crc16_byte(val.charAt(li),crc_inicial);
                }

                return crc_inicial;
        }

        public String crc16_avlgsm201(String val, int crc_inicial){
                // Função que chama a funcao crc16_string passando o comando como parametro e o
crc_inicial=0
                // E transforma o resultado de inteiro para HEX em formato string, exemplo: 10 =
"000A"
                String str;
                str = Integer.toHexString(crc16_string(val,crc_inicial));

                // Completa o CRC com zeros, caso ele não tenha 4 caracteres
                for (int li = str.length(); li < 4; li++){
                        str = "0".concat(str);
                }
                str = str.substring(2,4) + str.substring(0,2);
                str.toUpperCase();

                return (str.trim().toUpperCase());                //garante caixa alta
        }
}
```

AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.calcula_crc"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="14" />

   <application
         android:allowBackup="true"
         android:icon="@drawable/ic_launcher"
         android:theme="@style/AppTheme" >
         <activity
                android:name=".MainActivity">
                <intent-filter>
                        <action android:name="android.intent.action.SEND"/>
                        <category android:name="android.intent.category.DEFAULT"/>
                        <data android:mimeType="text/plain"/>
                </intent-filter>
         </activity>
   </application>
</manifest>
```