UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

HIDERALDO LUIZ ZANETTI

PROTÓTIPO PARA CONTROLE DE POSICIONAMENTO DE ANTENAS DIRECIONAIS UTILIZANDO DISPOSITIVOS MÓVEIS

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2013

HIDERALDO LUIZ ZANETTI

PROTÓTIPO PARA CONTROLE DE POSICIONAMENTO DE ANTENAS DIRECIONAIS UTILIZANDO DISPOSITIVOS MÓVEIS

MONOGRAFIA DE ESPECIALIZAÇÃO

Monografia de especialização apresentada ao Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná como requisito parcial para a obtenção do título de “Especialista em Tecnologia Java e Programação para Dispositivos Móveis”.

Orientador: Prof. Dr. João Alberto Fabro

CURITIBA

2013

AGRADECIMENTOS

Primeiramente agradeço a Deus pela oportunidade concedida de realizar esta nova conquista.

A todos os professores da UTFPR pela dedicação de tempo e esforço para transmitir seus conhecimentos com qualidade em um excelente ambiente para formação de especialistas.

Ao pessoal da coordenação do curso que sempre esteve à disposição para garantir a qualidade do serviço da instituição.

Agradeço a minha família pela compreensão de minha ausência devido ao tempo dedicado ao período deste curso, e também pelo apoio nos momentos de dificuldades.

RESUMO

O objetivo deste trabalho foi a elaboração de um protótipo para uso com dispositivo móvel para controlar o posicionamento de uma antena direcional de radioamadorismo que substitua os rotores e controles convencionais. Devido ao alto custo dos rotores disponíveis comercialmente e a necessidade da utilização de várias antenas em uma estação de radioamador, é apresentada neste projeto uma alternativa que pode ser utilizada para controlar as antenas direcionais de tamanho pequeno e peso reduzido. Esta monografia apresenta uma solução de baixo custo, onde é utilizado um dispositivo móvel com sistema operacional Android e a plataforma microcontrolada Arduino, para realizar o controle de motores de passo. Um módulo de comunicação que usa tecnologia Bluetooth integra os dois subsistemas, permitindo o controle sem fio do posicionamento direcional de uma ou mais antenas.

Palavras-chave: Dispositivos móveis, Android, Bluetooth, Arduino, Motor de passo, Radioamadorismo, Antena, Rotor.

ABSTRACT

The objective of this work was the development of a prototype for use with mobile device to control the positioning of a directional antenna of amateur radio to replace the conventional rotators and controls. Due to the high cost of commercially available rotators and the need of using multiple antennas in an amateur radio station, is presented in this project an alternative that can be used to control directional antennas with small size and reduced weight. This monograph presents a low-cost solution, which uses a mobile device with Android operating system and the microcontrolled platform Arduino, to control some stepper motors. A communication module using Bluetooth technology integrates the two subsystems, allowing wireless control of the directional positioning of one or more antennas.

Keywords: Mobile devices, Android, Bluetooth, Arduino, Step motor, Amateur radio, Antenna, Rotator.

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

No final do século XIX surgiram as primeiras experiências com emissões de rádio. Estes experimentos chamaram a atenção de várias pessoas que, por curiosidade, começaram a fabricar seus próprios equipamentos e antenas de forma artesanal, dando início a um *hobby* conhecido mundialmente que é o radioamadorismo.

Dependendo da faixa de frequências em que se opera e das condições atmosféricas, os radioamadores fazem contatos locais, internacionais e até mesmo comunicações via satélite. Para cada faixa de frequência é necessária uma antena com dimensões específicas, e dependendo do caso opta-se por antenas horizontais, verticais, omnidirecionais, direcionais, monobanda ou multibanda. Isto impõe que uma estação de radioamador possua várias antenas. A antena direcional, como o próprio nome sugere, deve ser direcionada para a localização onde se deseja estabelecer contato, possibilitando um melhor desempenho por concentrar a energia irradiada naquela direção. Para automatizar o posicionamento da antena são utilizados rotores que são controlados por um dispositivo que fica próximo ao operador da estação (LABRE-PR, 2013).

Uma vez que este conjunto composto por rotor e controle apresenta um custo elevado e há a necessidade de utilizarmos um para cada antena direcional, é proposta neste projeto uma alternativa de menor custo e com a possibilidade de ser utilizada para mais de uma antena. Através de um único dispositivo móvel (celular ou *tablet*) que trabalhe em conjunto com uma plataforma microcontrolada, será possível acionar motores de passo que permitirão o controle do posicionamento das antenas, trazendo praticidade e garantindo uma significativa redução no custo da estação. Para tornar o projeto ainda mais acessível, será feito uso de tecnologias abertas como o sistema operacional Android e da plataforma Arduino, que respectivamente fazem uso dos conceitos de *software* e *hardware* livres.

1.1. OBJETIVOS

1.1.1. Objetivo Geral

Elaborar um protótipo para uso com dispositivo móvel para controlar o posicionamento de uma antena direcional que substitua os rotores e controles convencionais.

1.1.2. Objetivos Específicos

- Definir qual o sistema operacional será utilizado no dispositivo móvel;
- Elaborar um *software* para controle de posicionamento de antenas que possa ser instalado em dispositivos móveis, no qual o usuário entrará com a informação de qual antena posicionar e de qual posição deseja que a antena adote;
- Fazer uso de uma técnica de comunicação sem fio para o sistema interagir com a interface controladora;
- Definir uma interface controladora para o motor de posicionamento;
- Desenvolver o sistema embarcado da interface controladora;
- Utilizar um motor de passo para controlar a posição da antena direcional.

1.2. JUSTIFICATIVA

De acordo com a legislação da Anatel, algumas das faixas de frequência de radioamador estão compreendidas entre 144 MHz (2 metros) e 440 MHz (0,7 metros), com isto pode-se concluir que nestas faixas as antenas são de dimensões reduzidas e de pouco peso, tornando possível a utilização de alternativas ao uso de um rotor disponível comercialmente, pois como todos os equipamentos para radioamadorismo os rotores encontrados no mercado são muito caros.

Geralmente uma estação de radioamador possui várias antenas e isto demanda vários rotores com seus controles individuais, o sistema proposto possibilitará substituir estes equipamentos por um único controle que será um celular ou *tablet*, reduzindo consideravelmente o custo da estação.

A escolha pelo sistema operacional Android se deve ao fato deste já ser o mais popular entre os utilizados em dispositivos móveis. A existência de uma vasta bibliografia, documentação, exemplos e bibliotecas de programas para integração com outros dispositivos garante que o sistema Android seja uma excelente opção para este projeto. Como demonstra a Figura 1, o Android além de ser o sistema operacional mais utilizado atualmente em dispositivos móveis, o número de novos usuários cresce a cada ano.

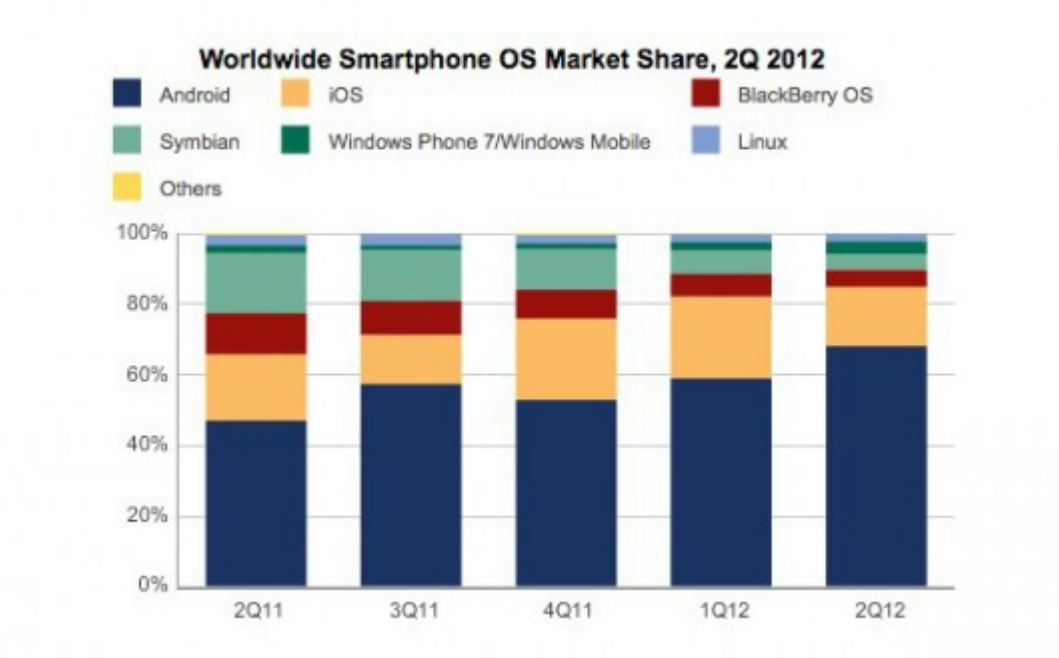

Figura 1 - Mercado mundial de Sistemas Operacionais para Smartphone

Fonte: CANALTECH (2013).

1.3. ORGANIZAÇÃO DO TRABALHO

No capítulo 2 é apresentada uma revisão da literatura dos temas relacionados ao desenvolvimento do trabalho que são: equipamentos de posicionamento de antenas comercializados; radioamadorismo; sistema operacional Android utilizado em dispositivos móveis; linguagem de programação para Android; sistemas de comunicação sem fio; interface controladora do motor de posicionamento; sistemas microcontrolados; motores de passo; técnicas de posicionamento de antenas.

No capítulo 3 são apresentados os detalhes do desenvolvimento do protótipo: diagrama de blocos; diagrama de classes e de casos de uso; telas e ícone do sistema móvel; fluxograma; esquema elétrico. Também será apresentado neste capítulo, como funcionam os subsistemas atuando de forma integrada.

Finalmente no capítulo 4 são apresentadas as conclusões, considerações e sugestões obtidas com a conclusão deste trabalho.

2. REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta os tópicos considerados relevantes para o entendimento das tecnologias utilizadas no projeto.

2.1. Ondas Eletromagnéticas

Como representada na Figura 2, uma onda eletromagnética é uma onda formada por campos elétricos e magnéticos que se propagam no espaço perpendicularmente um em relação ao outro. Maxwell, com base em experimentos de outros cientistas, estruturou um conjunto de equações que resume todos os conhecimentos sobre eletromagnetismo, as quais ficaram conhecidas como equações de Maxwell. Este conjunto de equações possibilitou a previsão da existência das ondas eletromagnéticas (BRASILESCOLA, 2013).

As ondas de rádio são ondas eletromagnéticas que se propagam na velocidade da luz, em todas as direções e podem percorrer grandes distâncias. Na atmosfera existem três camadas que influenciam de modos diferentes a propagação das ondas de rádio: a troposfera que é a camada mais baixa, a estratosfera que é a camada intermediária e a ionosfera que é a camada mais alta da atmosfera (PROTEVE, 2013).

As ondas de rádio ao atravessarem a troposfera mudam de forma ou de direção à medida que aumentam de altitude, isto devido ao aumento da velocidade em decorrência da atenuação dos gases. Algumas ondas de rádio podem refletir na troposfera e serem captadas por antenas que estejam fora da linha de visada da antena transmissora. A estratosfera não influencia significativamente a propagação das ondas de rádio e seus efeitos podem ser desconsiderados. Já a ionosfera que possui uma densidade de gases muito baixa e sofre um constante “bombardeio” de partículas e radiação solar, ionizando os gases que promove uma perda de potência muito lenta permitindo atingir distâncias muito maiores (C2O, 2013).

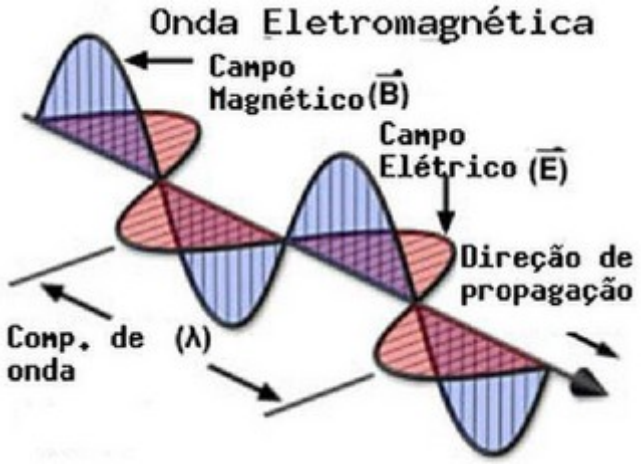

Figura 2 - Representação de uma Onda Eletromagnética

Fonte: C2O (2013).

2.2. Antenas

Segundo Balanis (1982, p. 1), antena é uma estrutura que faz a transição entre o espaço livre e um dispositivo guia. O dispositivo guia ou linha de transmissão pode ser um cabo coaxial ou guia de onda, e é usado para transportar energia eletromagnética de um transmissor para a antena ou da antena para um receptor.

Como a onda eletromagnética se propaga à velocidade da luz, ou seja, a 300.000 Km/s, para obtermos o máximo de eficiência de uma antena é necessário que haja uma relação correta entre o comprimento de onda e o tamanho da antena, esta relação é demonstrada na Figura 3. Para calcularmos o comprimento de onda, utilizamos a fórmula:

$$C = \frac{300.000.000}{f}$$

Onde:

C = comprimento da onda em metros (m)

f = frequência da onda em hertz (Hz)

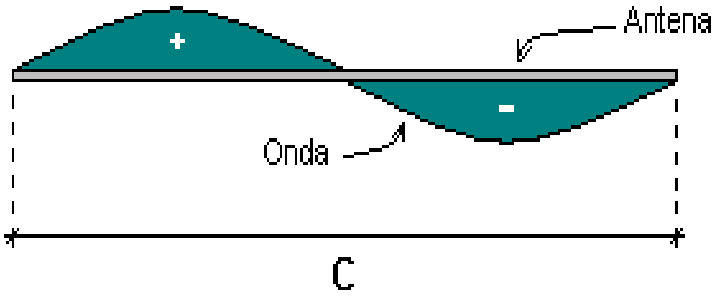

Figura 3 - Relação do tamanho da antena e comprimento de onda

Fonte: PROJTEC (2013).

A velocidade da onda eletromagnética na antena é 5% menor que no vácuo, então também devemos reduzir seu comprimento em 5%. Geralmente as antenas são construídas com seu tamanho em frações de comprimento de onda como, por exemplo, ½ onda e ¼ de onda.

Conforme Balanis (1982, p. 3), podemos construir antenas de diversas formas e tipos e devemos considerar alguns parâmetros fundamentais.

Antenas muito comuns, principalmente no radioamadorismo, são as antenas de fios e as antenas de arranjo do tipo Yagi-Uda, esta última por ser uma antena direcional e muito utilizada em frequências altas, será a antena abordada neste projeto (BALANIS, 1982).

De acordo com Balanis (1982), dentre os parâmetros fundamentais temos os padrões de irradiação, polarização, regiões de campo, densidade de potência, ganho, largura de banda e a diretividade da antena que é o mais importante para a aplicação deste projeto.

> “A diretividade de uma fonte não isotrópica é igual a taxa de sua intensidade de irradiação em uma determinada direção sobre a de uma fonte isotrópica.”
> (BALANIS, 1982, p. 39)

A antena direcional do tipo Yagi-Uda é formada por um número de elementos dipolos em linha, sendo um deles alimentado diretamente pela linha de transmissão e os outros agem como irradiantes parasíticos cujas correntes são induzidas por acoplamento mútuo (BALANIS, 1982, p. 513).

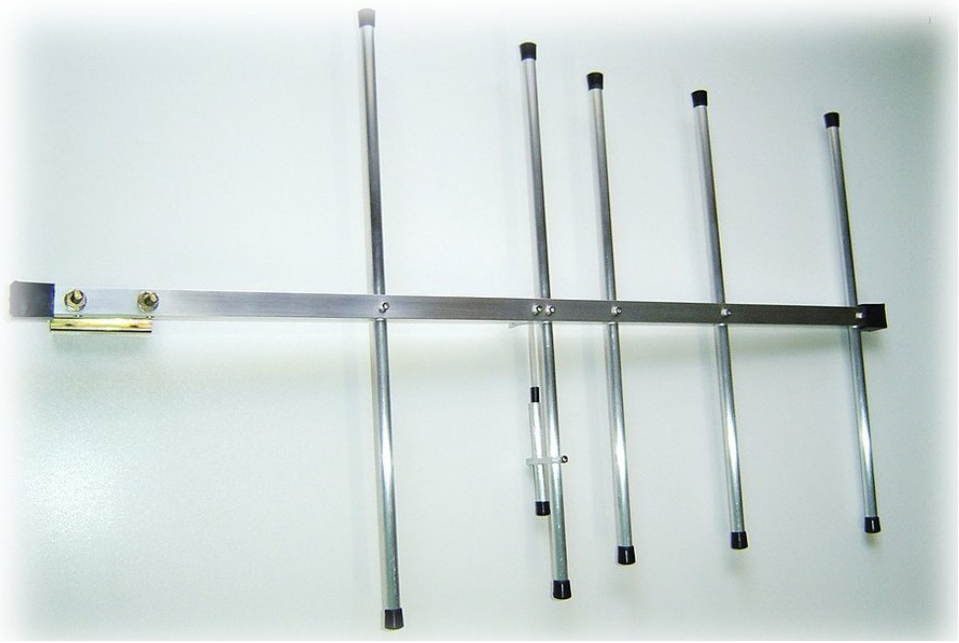

Figura 4 - Antena Yagi-Uda de 5 elementos

Fonte: ENGEHERTZ (2013).

2.3. Coordenadas Geográficas

No século XVII René Descartes, através de seus estudos relacionou álgebra com geometria e criou um sistema de coordenadas conhecido como plano cartesiano. Para estabelecer um ponto no plano cartesiano fazemos uso das coordenadas geográficas que são formadas por dois números que expressam sua localização na superfície da terra. Estes pontos são denominados latitude e

longitude. Para identificar estas coordenadas foram estabelecidas referências como a Linha do Equador, Paralelos e Meridiano.

A Linha do Equador é o círculo máximo perpendicular ao eixo de rotação da Terra e a divide em dois hemisférios, norte e sul, estabelecido como zero grau e é o início para a contagem da Latitude.

Os Paralelos são círculos menores paralelos à Linha do Equador, eles são os trópicos e os círculos polares.

O Meridiano de Greenwich divide a Terra em hemisfério oriental e ocidental e foi estabelecido como zero grau para contagem de Longitude.

A Longitude é a medida em graus de um ponto que parte do Meridiano de Greenwich, varia de 0° a 180° para leste ou para oeste.

A Latitude é a medida em graus de um ponto que parte da Linha do Equador e varia de 0° a 90° para norte ou para sul.

Figura 5 - Representação do plano cartesiano do mapa mundi

Fonte: NOT1 (2013).

2.4. Azimute

Azimute de um ponto é a medida em graus que indica um ponto no horizonte, é contada a partir do norte e no sentido horário. Esta medida varia de 0° a 360°. A referência de norte que utilizamos para posicionamento de antenas é o polo norte geográfico ao invés do magnético.

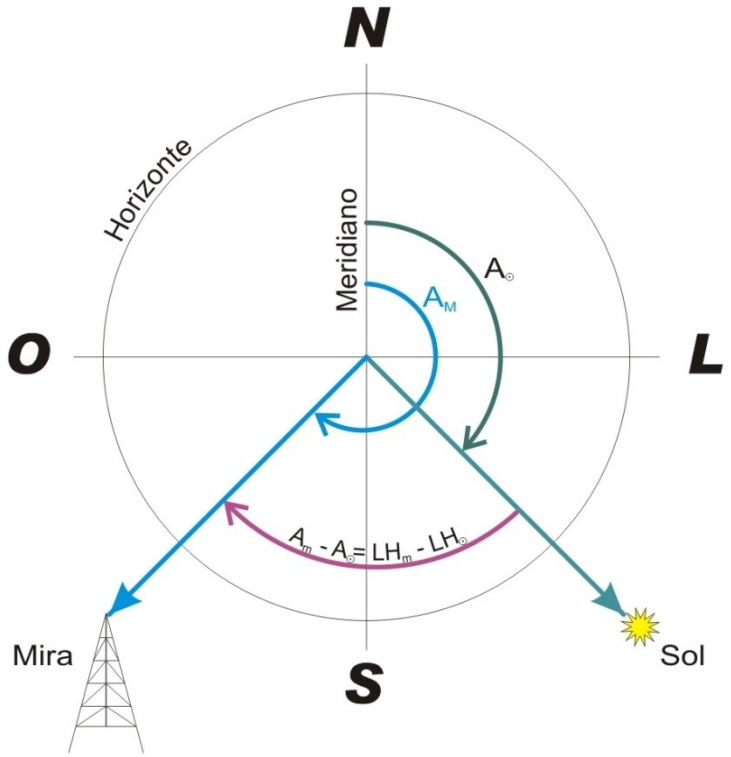

Figura 6 - Azimute

Fonte: UFRGS (2013).

2.5. Declinação Magnética

É o ângulo formado entre o norte verdadeiro (geográfico) e o norte magnético. Varia com o tempo e com a posição geográfica, pode ser ocidental ou

negativa quando o polo magnético estiver a oeste e oriental ou positiva quando estiver a leste.

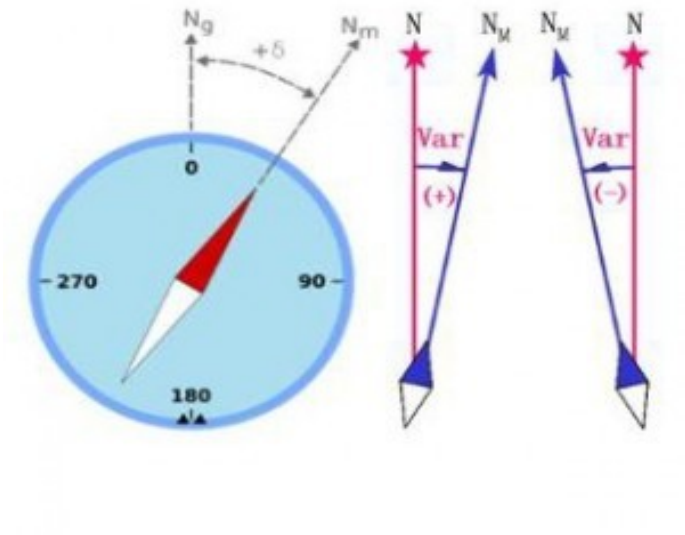

Figura 7 - Declinação magnética

Fonte: GLOBALTOP (2013).

2.6. Rotores de Antena

O rotor é um equipamento projetado para ser instalado no topo de mastros ou torres de antenas e são utilizados para direcionar uma antena para a posição desejada. Possui um controle de azimute que fica junto ao operador da estação, pode posicionar uma antena de 0° até 360°. São equipamentos muito resistentes, a prova d'água e projetados para suportar grandes e pesadas antenas. São responsáveis por aumentar em muito o custo de uma estação que utilize antenas direcionais. Seu preço varia de acordo com sua capacidade, no mercado nacional custam em torno de R$ 3.000,00.

Figura 8 - Rotor Yaesu G-1000DXA

Fonte: YAESU (2013).

2.7. Radioamadorismo

O radioamadorismo é um hobby científico praticado por pessoas licenciadas e que possuem conhecimentos técnicos em eletrônica, física, geografia e ainda outras áreas dependendo da modalidade utilizada, como radiotelegrafia, modos digitais e satélites.

Radioamador é a pessoa devidamente habilitada por órgãos reguladores, no Brasil o órgão regulador é a Anatel – Agencia Nacional de Telecomunicações. Para obter a licença inicial para montagem de uma estação de radioamador, denominada “classe C”, é necessário prestar exames de legislação de telecomunicações e ética operacional, com a aprovação nestes exames o radioamador estará habilitado a utilizar algumas faixas do espectro de frequências. Posteriormente o radioamador com mais de 18 anos de idade poderá prestar exames de conhecimentos básicos de eletrônica e eletricidade e transmissão e recepção auditiva de sinais em código Morse para obter o direito de operar em faixas adicionais, permanecendo um ano como “classe B” e depois automaticamente passando para “classe A”.

Uma estação de radioamador é composta por transceptores, fontes, amplificadores de potência, acopladores de antena, manipuladores de telegrafia e antenas diversas, equipamentos na sua grande maioria importados.

Além de ser uma atividade para troca de conhecimentos técnicos o radioamadorismo tem um papel muito importante em casos de calamidades públicas onde o radioamador presta uma grande colaboração trabalhando em conjunto com a defesa civil suprindo a deficiência nas comunicações causadas, por exemplo, por ventos e enchentes.

O site da Anatel define: “O Radioamador é o serviço de telecomunicações de interesse restrito, destinado ao treinamento próprio, intercomunicação e investigações técnicas, levadas a efeito por amadores, devidamente autorizados, interessados na radiotécnica unicamente a título pessoal e que não visem qualquer objetivo pecuniário ou comercial.”

Segundo relatórios do ano de 2012 da IARU – União Internacional dos Radioamadores (do original International Amateur Radio Union), atualmente são três milhões de radioamadores no mundo, abaixo é demonstrada uma tabela com os 11 países com a maior quantidade de estações licenciadas.

Posição	País	Estações Licenciadas
1	Japão	1.296.059
2	Estados Unidos	679.864
3	Tailândia	141.241
4	Coréia do Sul	141.000
5	Alemanha	79.666
6	China	68.692
7	Espanha	58.700
8	Reino Unido	58.426
9	Canadá	44.024
10	Rússia	38.000
11	Brasil	32.053

Figura 9 - Radioamadores no mundo

Fonte: DXBRASIL (2013).

2.8. Java

A linguagem de programação Java, foi desenvolvida por uma equipe da Sun Microsystems liderada por James Gosling. Java foi projetada em 1991 e seu nome original era Oak, inicialmente tinha o propósito de ser utilizada em sistemas embarcados em equipamentos eletrônicos. Em 1995 foi reprojetada para desenvolver sistemas para internet. Tornou-se muito popular e teve uma aceitação muito grande devido a sua característica de poder ser utilizada em qualquer plataforma. Como caracterizada pela Sun, Java é uma linguagem simples, orientada a objetos, distribuída, interpretada, robusta, segura, de arquitetura neutra, portável, de alto desempenho, multitarefa, e dinâmica. Java é rica em recursos, é uma linguagem de programação de uso geral, pode ser usada para desenvolver aplicações para missões robustas e críticas (LIANG, 2011, p. 8).

2.9. Sistema Operacional Android

Como observa Lecheta (2009, p. 20), o sistema operacional Android é uma proposta da empresa Google para ocupar o espaço existente no mercado de tecnologia móvel. Juntamente com o Google está o grupo Open Handset Alliance (OHA) que é composto pelas empresas líderes do mercado de telefonia como a Motorola, LG, Samsung, Sony Ericsson e outras. Este grupo foi criado com o intuito de padronizar uma plataforma de código aberto e livre para celulares para atender as expectativas e tendências do mercado atual. O Android é um sistema operacional baseado em Linux e seus componentes são escritos em linguagem C ou C++, já as aplicações desenvolvidas para Android são escritas em Java.

> “O Android tem muitos diferenciais interessantes e uma arquitetura realmente flexível focada na integração de aplicações. Não existe diferença entre uma aplicação nativa e uma desenvolvida por você.”
>
> (LECHETA, 2009, p. 23)

As aplicações em Android são construídas em linguagem Java, mas em seu sistema operacional não existe uma máquina virtual Java (JVM). O que temos é uma máquina virtual denominada Dalvik. Dalvik é uma máquina virtual otimizada para execução de aplicações em dispositivos móveis. As aplicações são desenvolvidas utilizando a linguagem Java e todos os seus recursos normalmente, depois o bytecode (.class) é compilado para o formato .dex (Dalvik Executable) que representa o aplicativo compilado. Após este processo os arquivos .dex juntamente com outros arquivos são compactados para um único arquivo do tipo .apk (Android Package File), que representa a aplicação final que será distribuída e instalada (LECHETA, 2009, p. 24).

Podemos utilizar nosso ambiente de desenvolvimento preferido, como o Eclipse e o Netbeans, para desenvolver nossos aplicativos para o Android. O Eclipse é o ambiente preferido pelo Google, e existe um plug-in chamado ADT (Android Development Tools) que facilita o desenvolvimento, testes e a compilação. Com o plug-in ADT executamos o emulador do Android diretamente do Eclipse o que nos permite controlar o emulador, utilizar os recursos do debug passo a passo, visualizar logs, simular envio de SMS ou ligação telefônica (LECHETA, 2009, p. 29).

2.10. Ciclo de vida do Android

Como explica Darwin (2012, p. 41), os aplicativos Android não tem um método “*main*”, é necessário entender como eles começam, param ou são parados. O aplicativo executa seu próprio processo Unix, não afetando nenhum outro aplicativo em execução. A máquina virtual Dalvik se encarrega de fazer a interface

com o sistema operacional para chamar você quando o aplicativo é iniciado ou quando houver mudança de aplicativo pelo usuário. Como é demonstrado na Figura 10, são possíveis três estados para um aplicativo Android:

- Running: Ativo, o aplicativo está em execução e visível para o usuário.
- Paused: Pausado, o aplicativo está em execução, mas perdeu o foco de entrada e está parcialmente obscurecido.
- Stopped: Parado, o aplicativo está em execução, mas completamente oculto à visão do usuário.

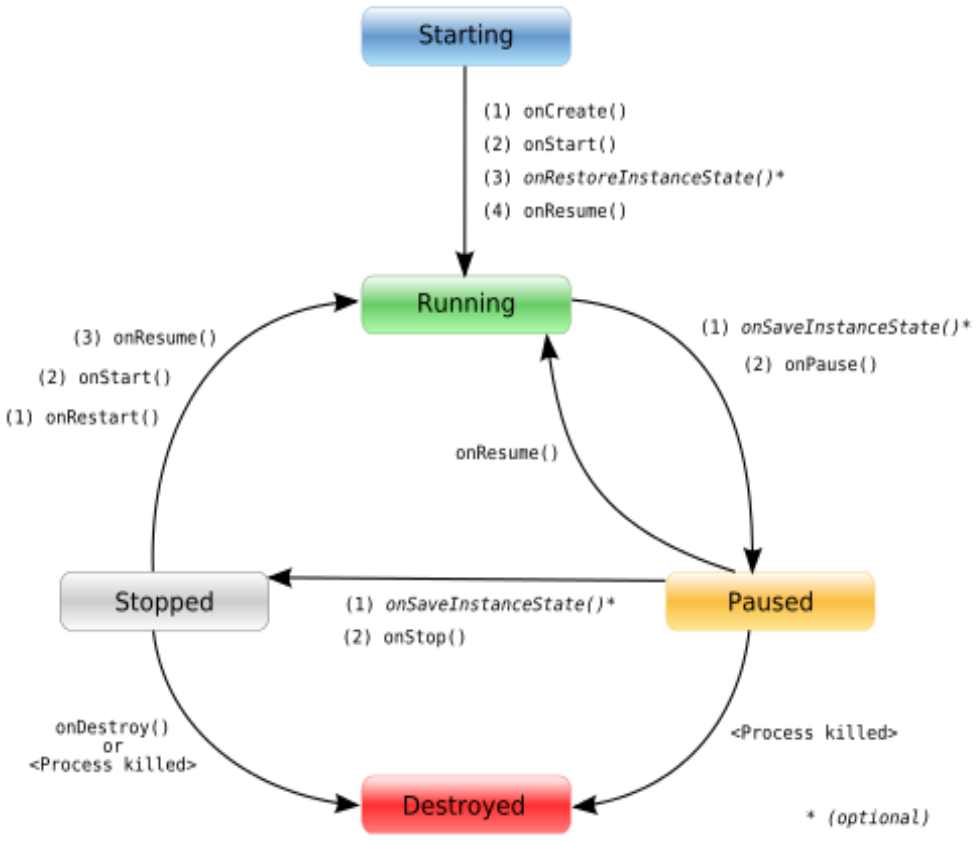

Figura 10 - Ciclo de vida de uma aplicação Android

Fonte: JAVAHISPANO (2013).

2.11. Arduino

O Arduino Project, que é baseado no conceito de tecnologia aberta, foi iniciado em 2005, e mais de 150 mil placas foram vendidas em todo o mundo. Suspeita-se que a comercialização de placas clone não oficiais superam este número. A popularidade do Arduino não pára de crescer, seu potencial atrai cada vez mais as pessoas que procuram uma solução prática e de baixo custo para seus projetos, e ainda possui uma curva de aprendizagem muito pequena (MCROBERTS, 2011).

McRoberts (2011, p. 22) define o projeto Arduino como um pequeno computador que pode ser programado para processar sinais de entrada e saída, permitindo que ele interaja com os componentes externos a que está conectado. Chamamos o Arduino de plataforma de computação física ou embarcada, isto é, um sistema que interage com seu ambiente através de software e hardware. O Arduino pode ser utilizado para desenvolver objetos interativos independentes, e também é possível conectá-lo a um computador, rede e internet (MCROBERTS, 2011).

A plataforma Arduino é composta de uma placa com um processador Atmel, um cristal ou oscilador e um regulador de tensão de 5 volts. Dependendo do modelo, pode ter uma saída USB que pode ser utilizada para conectá-lo a um computador. A placa apresenta pinos de entrada e saída para serem utilizadas por outros circuitos ou sensores. Para programar o Arduino, na linguagem que ele compreende (baseada em C), temos disponível como software livre o IDE do próprio Arduino (MCROBERTS, 2011).

Para aumentar seu potencial o Projeto Arduino ainda conta com extensões denominadas s*hields*, que são placas de circuito que contém outros dispositivos que podem ser conectadas a ele para obter funcionalidades adicionais. Temos *shields* com a função de GPS, Ethernet, Bluetooth, WI-FI e outros mais.

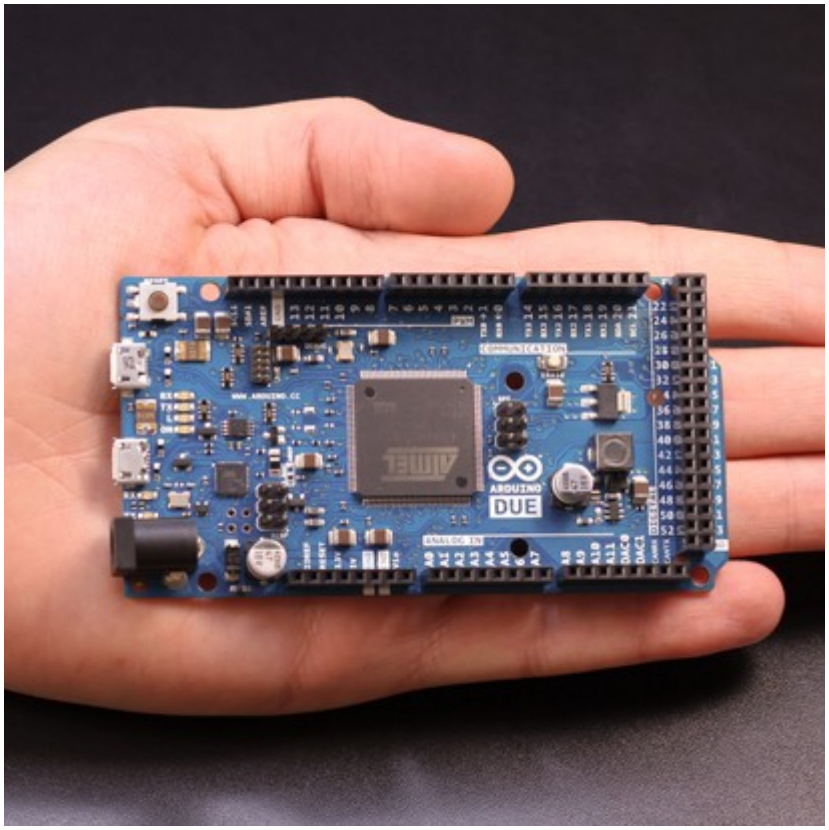

Figura 11 - Placa Arduino

Fonte: ARDUINO (2013).

2.12. Redes sem fio

As redes sem fio foram criadas para complementar as redes cabeadas, dando mobilidade e agilidade ao acesso a dados independentemente do lugar onde o usuário esteja. Os dados são transmitidos usando o ar ou espaço livre como meio físico permitindo uma grande flexibilidade na localização das estações (TELECO, 2012).

A utilização de protocolos de redes sem fio vem crescendo muito tanto para uso doméstico como na indústria. Existem inúmeros protocolos que podem ser aplicados desde que sejam levados em conta fatores como consumo de energia, sensibilidade, alcance, segurança e taxas de transferência. Devido a redução de custos e sua flexibilidade em comparação com as redes cabeadas, protocolos como Zigbee, Bluetooth e WI-FI vem se destacando (TELECO, 2012).

A Figura 12 representa um comparativo entre algumas tecnologias de redes sem fio. Pode-se verificar que a tecnologia ZigBee e Bluetooth têm praticamente o mesmo alcance, mas o Bluetooth a supera em velocidade. Já uma rede Wireless tem um alcance e taxa de transmissão superiores ao Bluetooth e ZigBee. Mesmo estando no gráfico as tecnologias UWB e 2.5G / 3G não foram objetos de estudo para desenvolvimento deste trabalho.

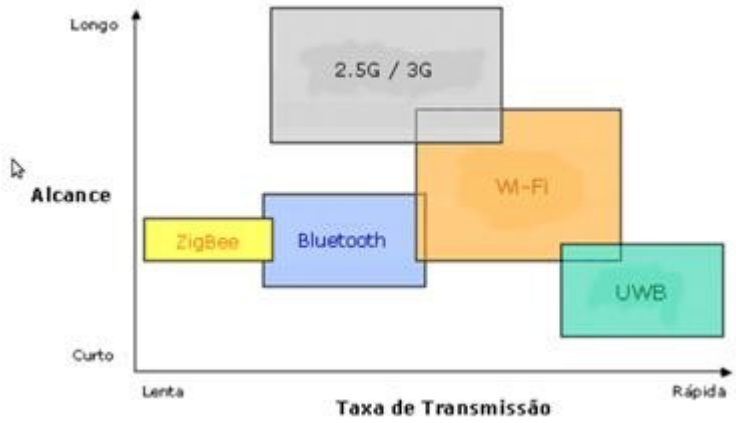

Figura 12 - Alcance de redes sem fio

Fonte: TELECO (2013).

2.13. Motor de Passo

Motores de passo são dispositivos eletromecânicos que fazem a conversão de pulsos elétricos em movimentos mecânicos que geram variações angulares discretas (UFF, 2008).

São amplamente utilizados em aplicações quando movimentos precisos são necessários, pois permitem controlar fatores como ângulo de rotação, velocidade, sincronismo e posição. Seu funcionamento é dado por solenoides

alinhados dois a dois que quando energizados alinham o rotor causando uma variação angular que é chamada de passo (UFF, 2008).

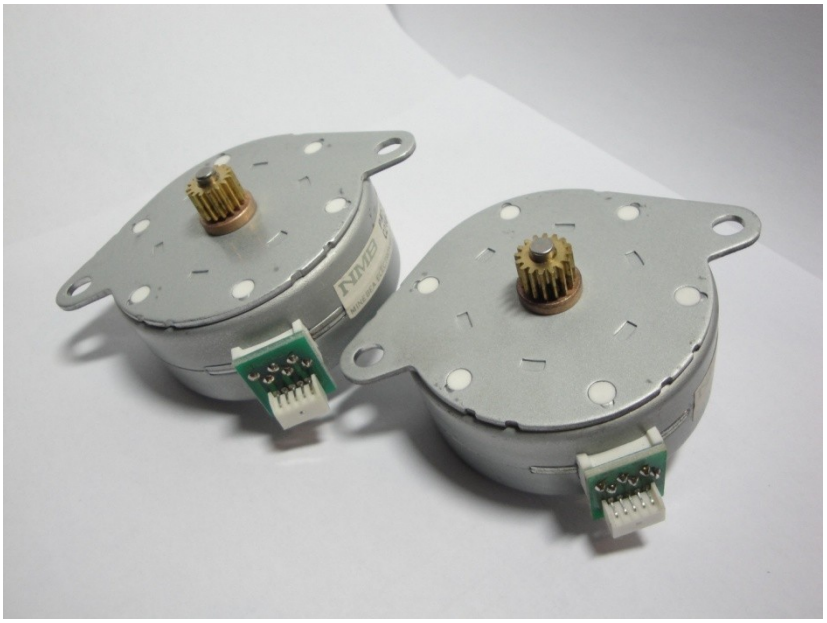

Figura 13 - Motores de passo PM55L-048

Fonte: ROBOCORE (2013).

Os motores vistos na Figura 13, também podem ser encontrados em impressoras do tipo jato de tinta.

3. DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo, é apresentado o protótipo desenvolvido para este projeto agora denominado BTAC - Bluetooth Antenna Controller. São demonstrados o diagrama de casos de uso, as classes criadas para o sistema, as imagens das telas do aplicativo e os circuitos elétricos utilizados no protótipo do rotor. Também é explanado o funcionamento integrado dos módulos do sistema. Devido ao sistema de controle ter sua aplicação e comportamento bem específicos, neste projeto não se aplica a especificação de requisitos funcionais e não-funcionais.

3.1. Aplicativo Móvel

O aplicativo móvel foi desenvolvido para ser implantado em um dispositivo que utilize o Android como sistema operacional. Fará uso da interface Bluetooth como meio de comunicação com o sistema embarcado na plataforma Arduino. No início da execução do aplicativo é apresentada uma tela com a informação do nome e versão do sistema, e que também possibilita ao usuário conectar com o dispositivo Bluetooth externo. Depois de conectado o aplicativo permite ao usuário optar entre duas antenas a serem controladas. Na tela é exibida a posição em que cada antena se encontra no momento, e o usuário pode informar qual a nova posição desejada para a antena selecionada. Também está disponível uma opção de inicialização, a qual faz com que a antena procure automaticamente pela posição “zero”, de acordo com o sensor instalado junto ao seu rotor. Isto pode ser útil quando a antena tiver sua posição alterada sem que tenha sido pelo sistema controlador. Maiores detalhes sobre a atuação do sensor são descritos na seção 3.6.

Outra característica do aplicativo é que foram implementadas as técnicas de internacionalização para os idiomas português e inglês. O idioma padrão do aplicativo é inglês, mas se o sistema operacional estiver configurado para português as telas e mensagens serão apresentadas neste idioma.

3.1.1. Diagrama de Casos de Uso

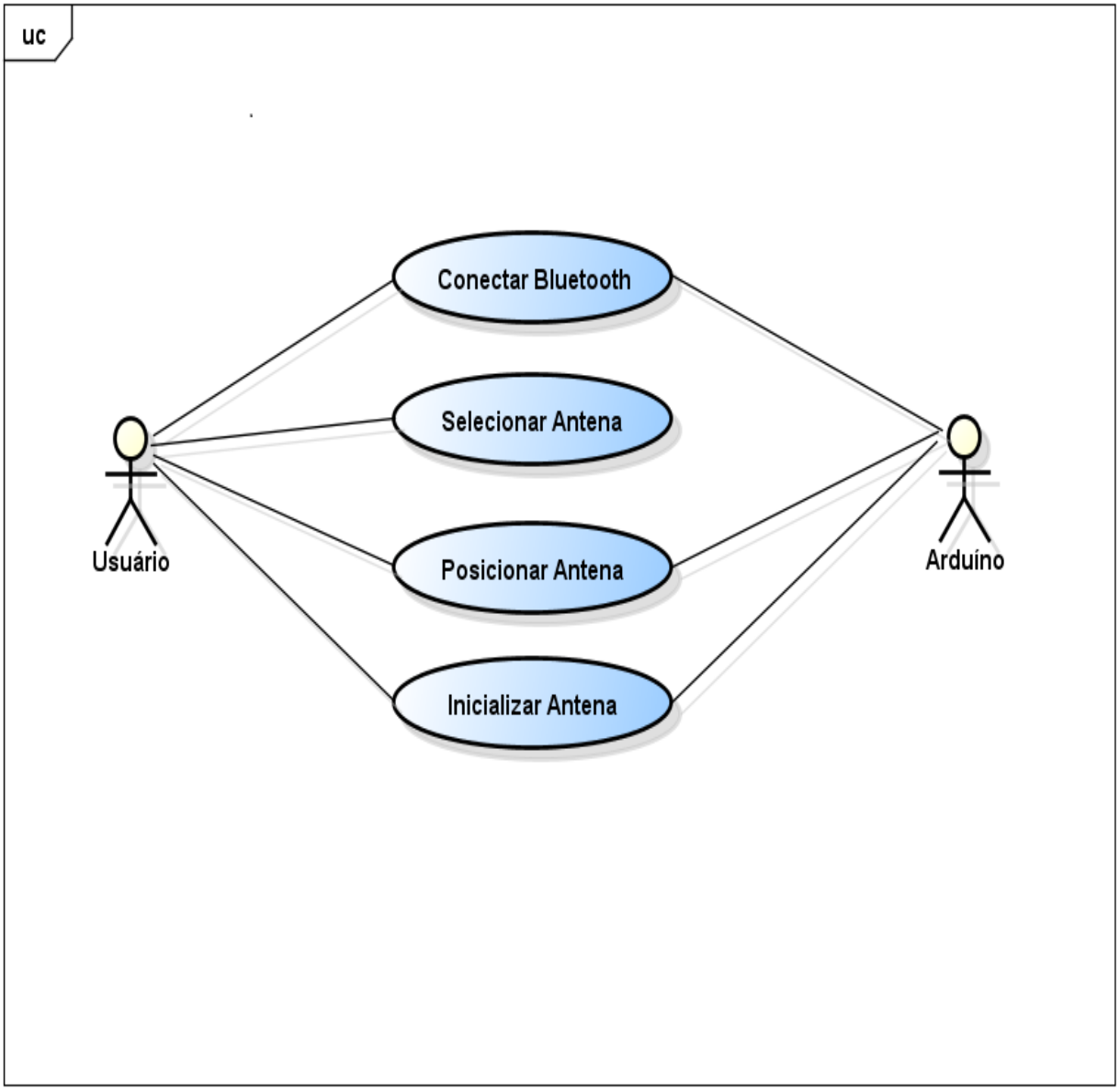

Figura 14 - Diagrama de Casos de Uso

Fonte: Autoria própria.

3.2.2. Descrição dos Casos de Uso

- Conectar Bluetooth: Este caso de uso é encarregado de identificar se o dispositivo móvel possui a funcionalidade de Bluetooth e se este serviço está ativo, também faz a conexão com o dispositivo Bluetooth externo (Arduino) cuja identificação já está previamente definida no aplicativo. O dispositivo externo já deve estar previamente “pareado” com o dispositivo móvel.

- Selecionar Antena: Tem a função de definir uma das antenas como sendo a que está em uso no momento, ou seja, para qual antena os comandos serão enviados.

- Posicionar Antena: Faz uso da informação que identifica a posição desejada para a antena corrente, esta informação é inserida pelo usuário no aplicativo móvel. Este caso de uso também faz a montagem do comando de posicionamento e o envia para o dispositivo externo através da conexão Bluetooth.

- Inicializar Antena: Este caso de uso envia, ao dispositivo externo, o comando que faz com que a antena busque pela sua posição “zero”. É utilizado caso o controle de posicionamento tenha sido perdido.

3.2.3. Diagrama de Classes

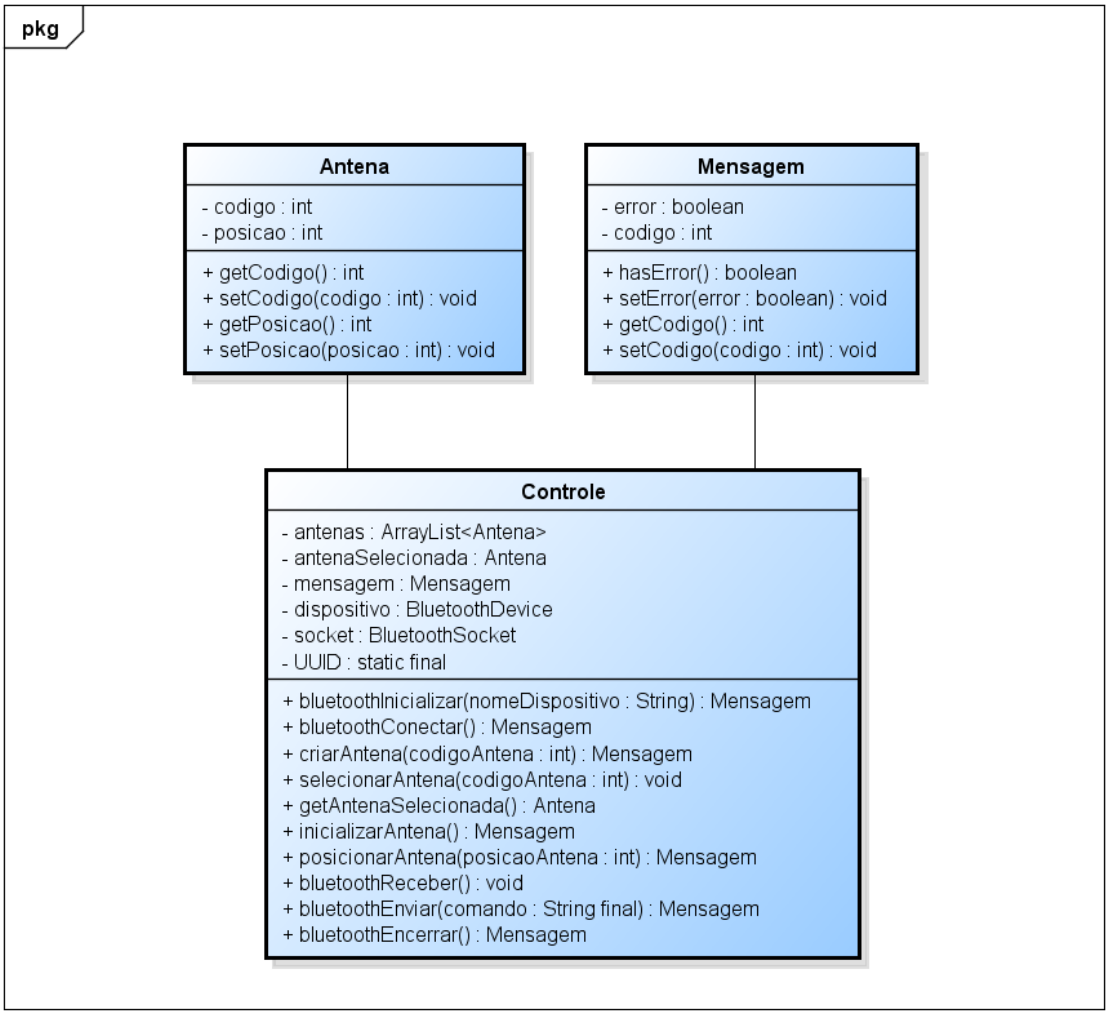

Figura 15 - Diagrama de Classes

Fonte: Autoria própria.

3.2.4. Ícone e Telas

O *kit* de desenvolvimento para Android possui uma grande variedade de bibliotecas que disponibiliza vários componentes para serem utilizados no desenvolvimento de programas. Estes componentes permitem que sejam criadas telas simples e intuitivas. Como exemplo pode ser visto na Figura 18, onde é apresentada a “Tela de Controle” que mesmo necessitando de digitação de números, é utilizada uma alternativa muito prática que dispensa o uso o teclado.

A Figura 16 demonstra o ícone que será apresentado na tela do dispositivo móvel após a instalação do aplicativo.

Figura 16 - Ícone

Fonte: Autoria própria.

Na figura 17 é pode-se ver a tela que será apresentada ao usuário ao executar o aplicativo, permitindo que este solicite a conexão com o sistema Arduino.

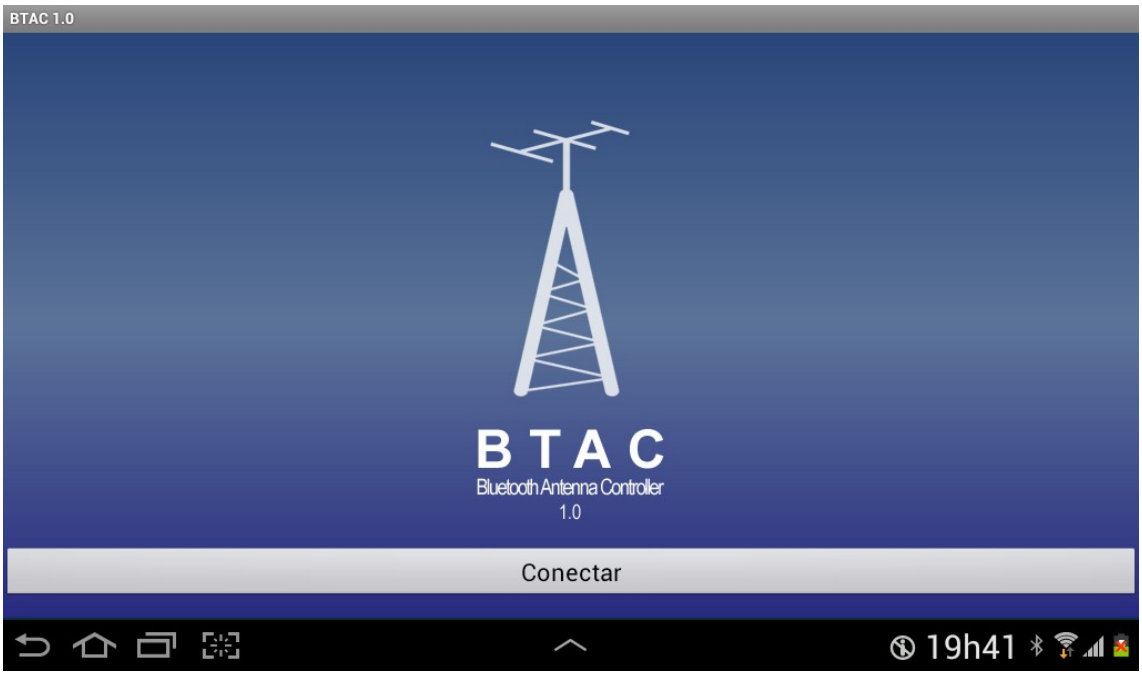

Figura 17 - Tela Inicial

Fonte: Autoria própria.

A tela de controle de posicionamento das antenas, onde o usuário escolhe qual antena controlar e qual a posição desejada, é apresentada na Figura 18.

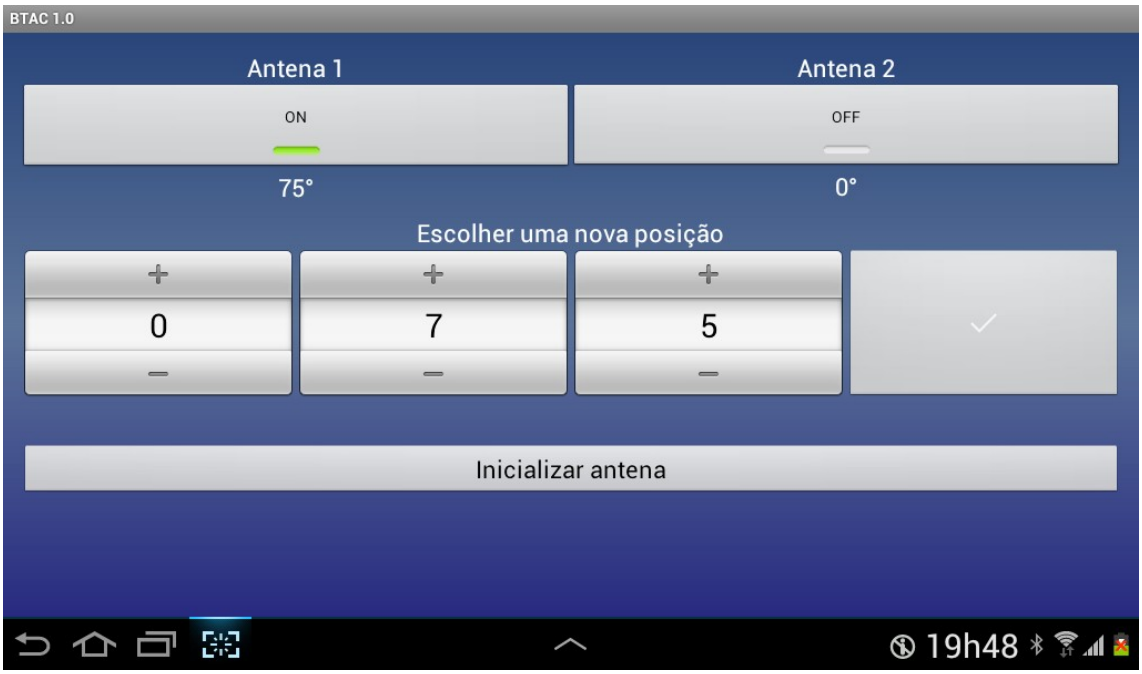

Figura 18 - Tela de Controle

Fonte: Autoria própria.

3.3. Sistema Embarcado

Para o desenvolvimento do sistema para a plataforma microcontrolada Arduino, foi utilizado seu próprio ambiente de programação, que permite a programação em linguagem C. Este ambiente também disponibiliza a compilação, depuração e a transferência do programa para o microcontrolador através de uma porta USB do computador.

O sistema embarcado recebe os comandos do dispositivo móvel através de comunicação Bluetooth. Para estes comandos foram definidos formatos que possibilitassem a identificação da ação a ser executada, código da antena a

controlar e também a informação do posicionamento desejado. Na sequência é demonstrado o fluxograma geral do sistema:

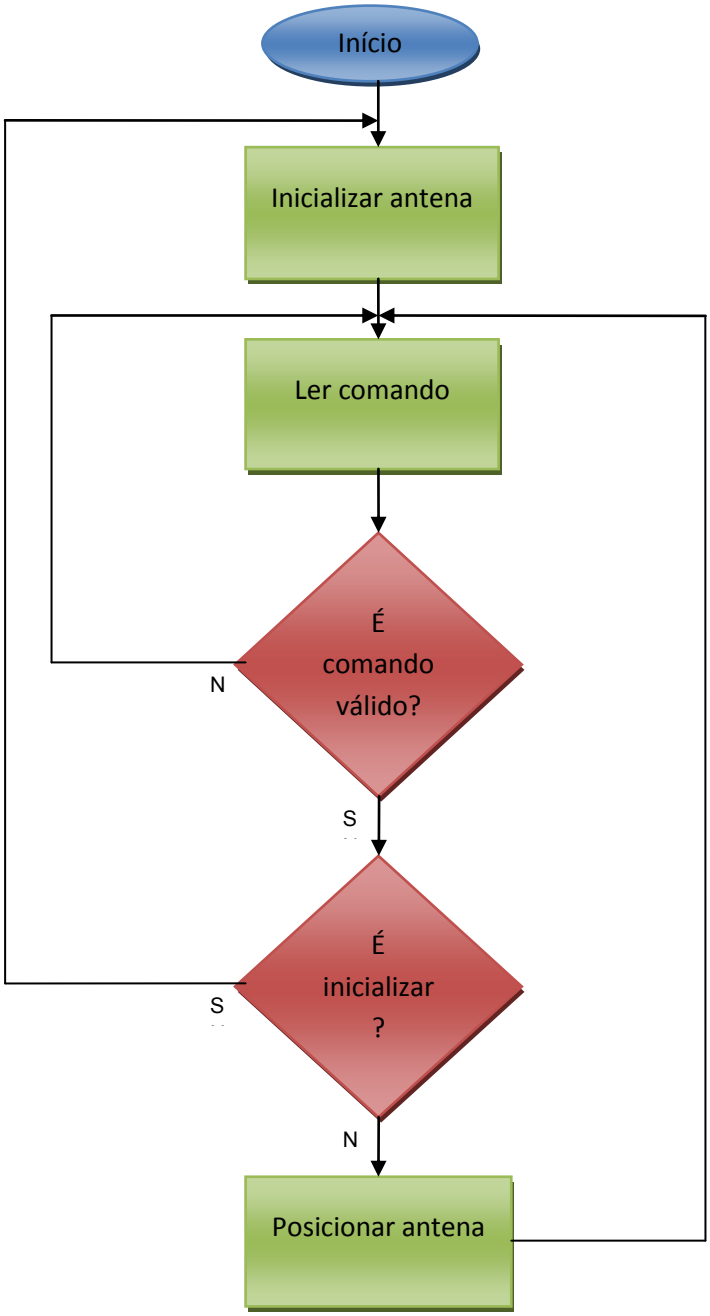

Figura 19 - Fluxograma do Sistema Embarcado

Fonte: Autoria própria.

3.4. Comunicação entre Android e Arduino

Após serem analisados alguns tipos de comunicação sem fio, foi optado pela comunicação via Bluetooth, pois este recurso pode ser encontrado em praticamente todos os dispositivos móveis. Também existem várias opções para adaptar o Bluetooth ao Arduino, com custo bem acessível e de fácil montagem. A figura abaixo é uma imagem do adaptador Bluetooth para Arduino utilizado neste projeto.

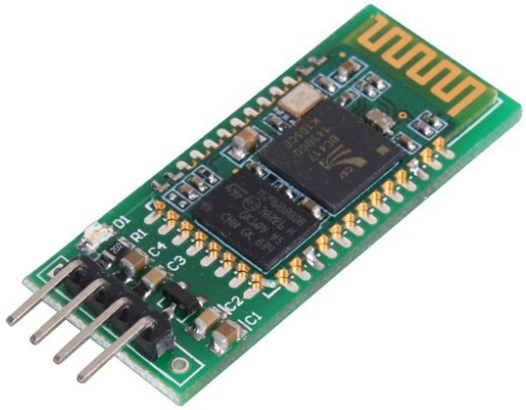

Figura 20 - Shield Bluetooth

Fonte: WITHUS (2013).

3.5. Motores de Passo

Durante a montagem do protótipo do rotor, constatou-se a necessidade da utilização de um *driver* para acionar os motores de passo, pois a corrente elétrica fornecida pelo Arduino não é suficiente para este trabalho. Foi utilizado o circuito integrado ULN-2003 como *driver*, sendo necessário um para cada motor utilizado, ou seja, para cada antena. Para garantir o posicionamento inicial do rotor em zero grau,

foi utilizado um foto-sensor modelo H21A1, que com a ajuda de um cursor instalado no eixo do rotor, envia um sinal ao Arduino, informando que o rotor está pronto para receber comandos de posicionamento. A Figura 21 mostra a placa auxiliar, na qual pode-se observar os dois drivers de motores de passo, o *shield* Bluetooth em seu suporte, as conexões para motores, Arduino, fonte e sensor.

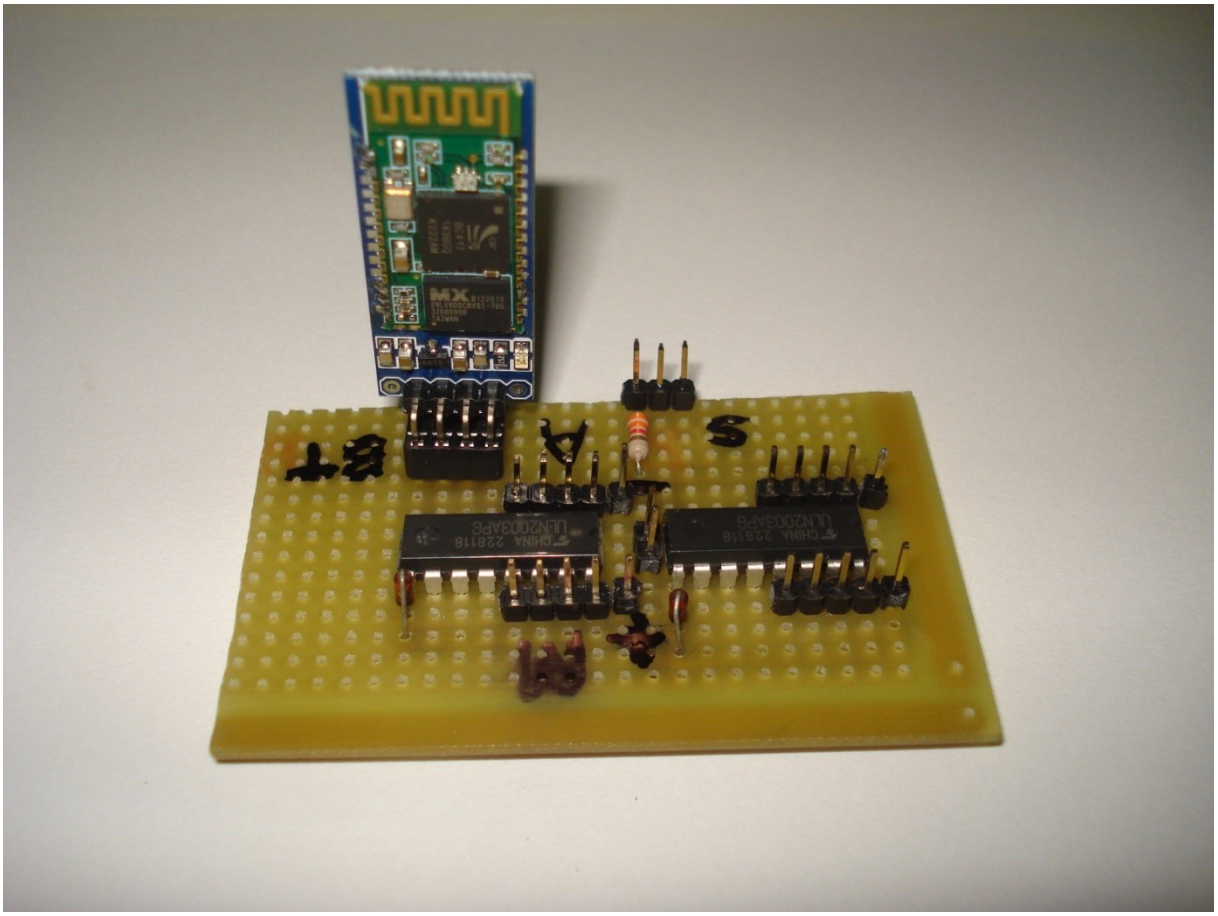

Figura 21 - Placa Auxiliar

Fonte: Autoria própria.

3.6. Sensor

O H21A1 é um sensor fotoelétrico, isto é, é um componente eletrônico que responde à variação de luz que incide sobre ele. É composto por dois componentes: um diodo emissor de luz e um transistor que detecta a luz emitida pelo diodo.

Figura 22 - Sensor H21A1

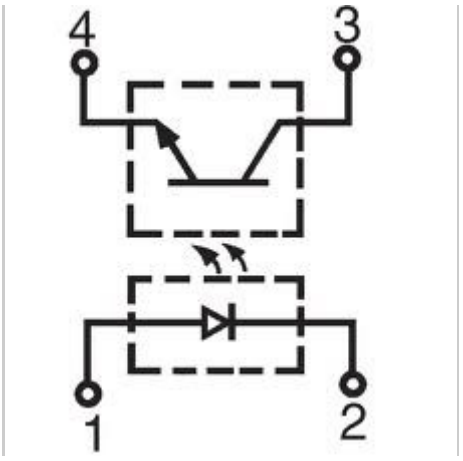

Figura 23 - Representação Elétrica do H21A1

Fonte: CHIPFIND (2013).

3.7. Funcionamento Integrado dos Módulos do Projeto

A Figura 24 mostra como cada módulo do projeto se relaciona, ilustrando o sentido em que as informações trafegam.

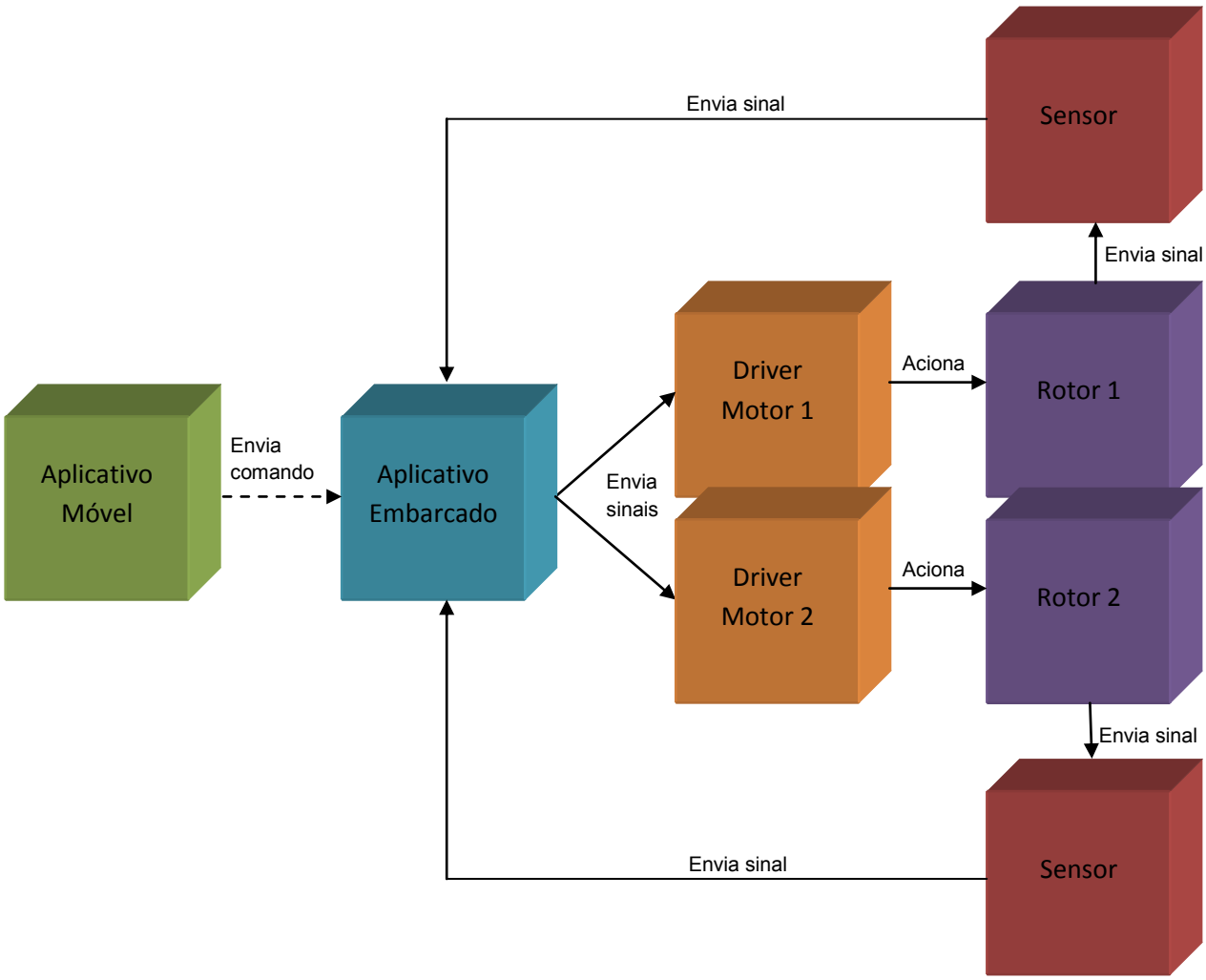

Figura 24 - Diagrama de Blocos do Projeto

Fonte: Autoria própria.

No aplicativo móvel o usuário escolhe a antena a controlar, informa a posição, em graus, desejada. Ao confirmar este comando, o aplicativo móvel

constrói a mensagem a ser enviada ao aplicativo embarcado. A construção desta mensagem segue o seguinte protocolo:

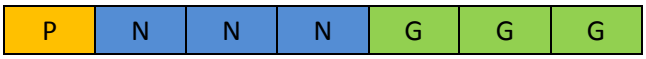

Figura 25 - Leiaute do Protocolo

Fonte: Autoria própria.

- P: Caractere fixo “P” para identificar que a informação é um comando;
- NNN: Identificador do código da antena selecionada. O conteúdo deve ser composto por três dígitos numéricos. Exemplo: 001;
- GGG: Parâmetro contendo a posição absoluta, em graus, solicitada pelo usuário. O conteúdo deve ser composto por três dígitos numéricos. Exemplo: 180. Para o comando de inicialização o conteúdo deste parâmetro é 999.

Exemplos:

1. Para posicionar a antena 1 em 180 graus a mensagem a enviar seria: P001180.
2. Para inicializar a antena 1 a mensagem seria: P001999.

Após a construção da mensagem, o sistema móvel a envia ao dispositivo embarcado através de comunicação Bluetooth.

O sistema embarcado monitora constantemente a porta serial para verificar a chegada de informações. Ao detectar que existem informações disponíveis, o sistema analisa seu conteúdo e verifica se é uma mensagem de comando conhecida. Caso seja identificado como um comando, o sistema extrai as informações da mensagem, obtendo o código da antena e da posição. Como este projeto foi elaborado para controlar até duas antenas, o sistema já tem predefinido quais os pinos do Arduino estão associados à antena em questão, e os utiliza para enviar os sinais elétricos para controle do motor. Estes sinais são enviados primeiramente a outro componente denominado *driver,* o qual tem a função de

amplificá-los antes de serem repassados ao motor. O sistema embarcado mantém o registro da posição de cada antena e então calcula a diferença entre a posição atual e a nova posição solicitada pelo usuário. Após este processo, envia somente os sinais que atualizam a direção da antena.

Um sensor, instalado juntamente com cada motor, monitora a posição da respectiva antena para garantir que esta adote a posição “zero” quando o sistema for iniciado. Isto é necessário porque o sistema pode ser desligado com a antena em qualquer posição, impossibilitando o controle no próximo uso. Ao iniciar o sistema as duas antenas executam o procedimento de inicialização, e quando a antena atinge a posição inicial correta, o sensor envia um sinal ao sistema embarcado, que identifica que a antena está pronta para uso. Caso o usuário perceba alguma anomalia no posicionamento da antena, ele pode solicitar o procedimento de inicialização através do botão “Inicializar Antena”.

A Figura 26 mostra como devem ser feitas as ligações elétricas entre os módulos, componentes eletrônicos, e também as tensões elétricas estabelecidas para cada um.

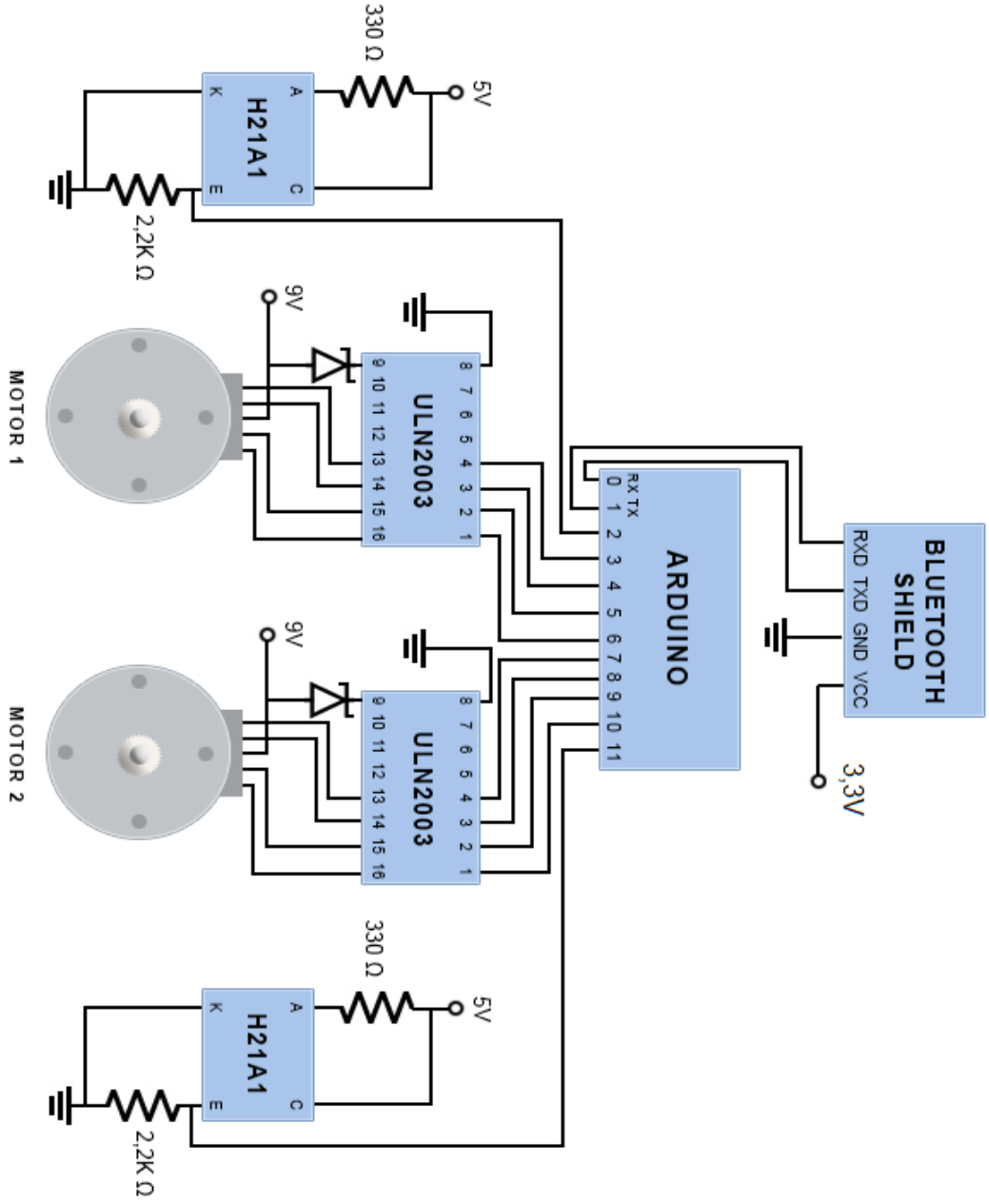

Figura 26 - Esquema Elétrico

Fonte: Autoria própria.

Na Figura 27 é mostrado como o sensor fotoelétrico é montado junto ao suporte do motor, de forma que o cursor possa passar livremente entre seus componentes e consiga obstruir a passagem da luz emitida pelo fotodiodo para que o sensor funcione como uma chave.

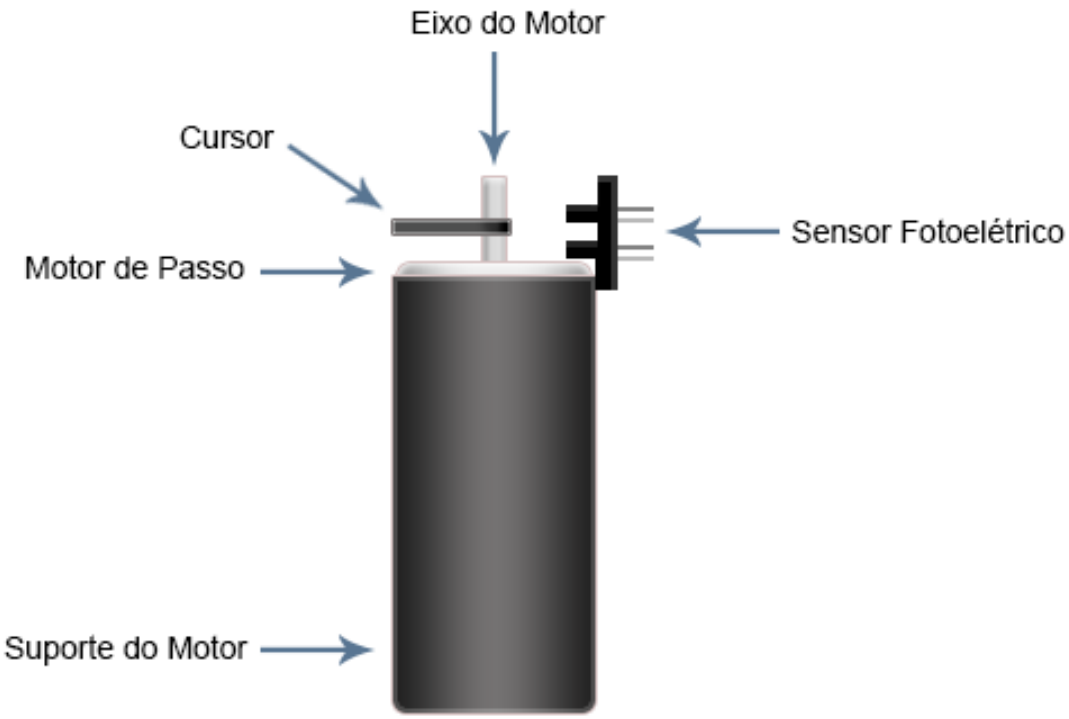

Figura 27 - Montagem do Rotor

Fonte: Autoria própria.

O conjunto completo que compõe o protótipo é apresentado na Figura 28, onde podem ser vistos: o Arduino, a placa com os *drivers* para os motores de passo, o rotor juntamente com o sensor montados em um pedaço de tubo de PVC, o cabo de rede CAT5 que interliga as placas ao rotor.

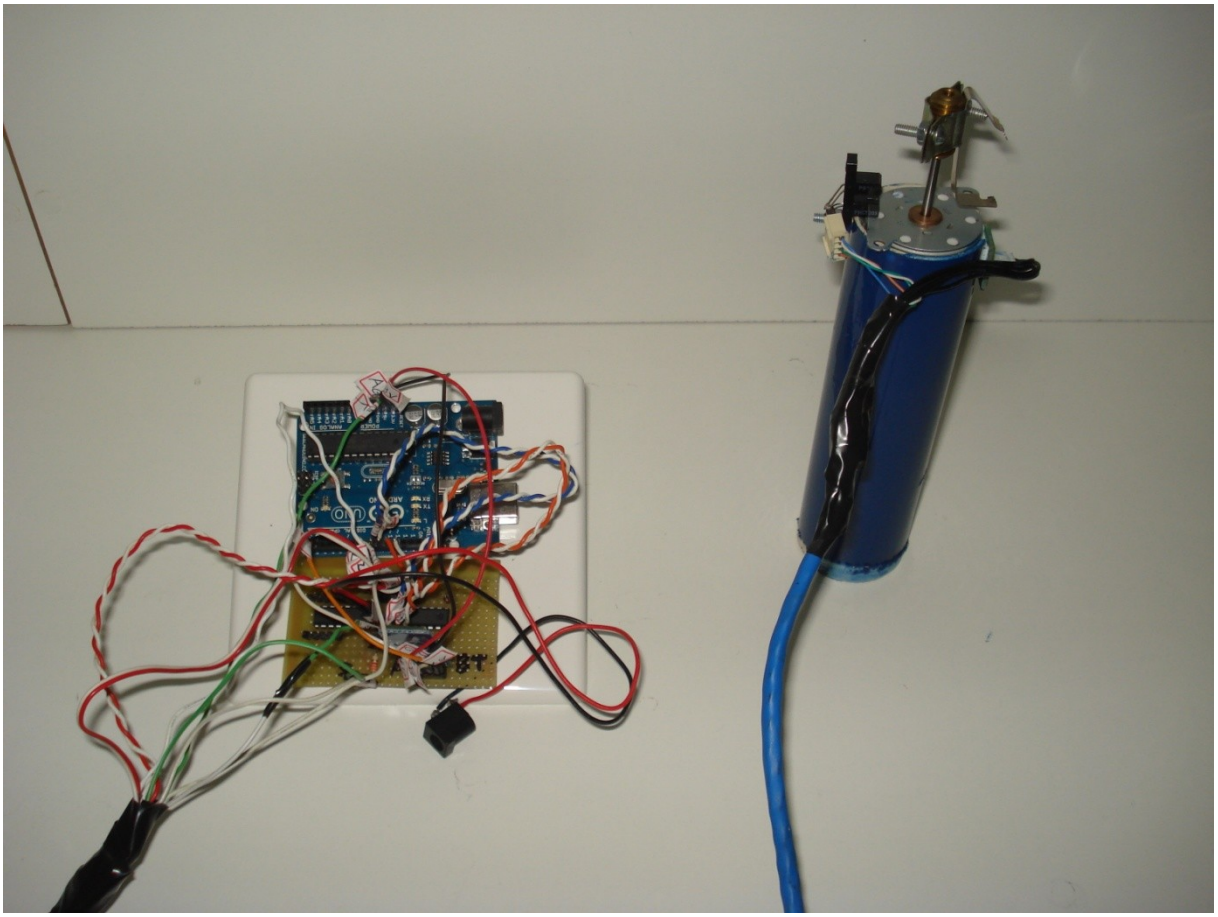

Figura 28 - Protótipo Completo

Fonte: Autoria própria.

4. CONCLUSÃO

Como visto em várias literaturas foi possível confirmar, durante o desenvolvimento deste projeto, o grande potencial de integração do sistema operacional Android com outros dispositivos. Alguns livros e consultas à internet foram suficientes para superar todas as dificuldades encontradas na fase de programação. As bibliotecas do Android de apoio ao desenvolvimento foram fundamentais para concretização do projeto, principalmente na etapa de comunicação utilizando dispositivo Bluetooth.

Como sugestão de implementações para o aplicativo, seria muito interessante a possibilidade de controlar várias antenas, onde o usuário poderia fazer um cadastro com quantas antenas desejasse. Para isto poderia ser utilizado algum tipo de banco de dados com uma tabela com as informações de cada antena. Consequentemente isto necessitaria de algumas implementações no hardware a ser utilizado em conjunto com o Arduino, mas o uso de circuitos lógicos poderia ser uma ótima solução. Outra melhoria sugerida, seria a de que neste mesmo banco de dados pudessem ser cadastrados os dados de localização do usuário e também da localização de outras estações, possibilitando ao usuário escolher a estação a contatar, e após, o sistema automaticamente calcularia a direção de apontamento e enviaria o comando de posicionamento à antena.

Algumas dificuldades apareceram durante a interligação dos módulos, pois como apresentado no esquema elétrico, o sistema utiliza componentes que exigem tensões elétricas específicas, o que exige a utilização de conexões distintas e, consequentemente, uma maior quantidade de fios. Como solução é sugerido um projeto melhor elaborado para a placa auxiliar, no qual podem ser incluídos reguladores de tensão para os casos específicos.

O protótipo de rotor mostrou ser uma opção interessante para alguns casos específicos de controle de antena direcional. Um exemplo seria ao montar uma estação temporária em que as condições climáticas estejam favoráveis, como a não existência de vento e chuva, pois um rotor de construção caseira, feito com motor de passo, poderia ser facilmente prejudicado por estes fatores. Para minimizar

a interferência do vento, devem-se manter algumas bobinas do motor de passo energizadas tendo como resultado um grande aquecimento do motor. Outro limitador é a altura em que a antena pode ser instalada, pois se deve observar a distância que os motores de passo estariam de seus *drivers,* neste projeto foram feitos testes com cabo de rede CAT5 a uma distância de 10 metros, e nestes testes o sistema comportou-se satisfatoriamente.

5. REFERÊNCIAS

ANATEL: **Radioamador**. Disponível em: <http://www.anatel.gov.br/Portal/exibirPortalNivelDois.do?codItemCanal=1220&nomeVisao=Cidad%E3o&nomeCanal=Comunica%E7%F5es%20Via%20R%E1dio&nomeItemCanal=Radioamador>. Acesso em 06 de mai.2013.

ARDUINO: **Projeto Arduino**. Disponível em: <http://www.arduino.cc/>. Acesso em 15 de abr.2013.

BALANIS, Constantine A., **Antenna theory: Analysis and Design.** 2ª ed., United States of America: Wiley, 1982.

BRASILESCOLA: **O que são ondas eletromagnéticas?**. Disponível em <http://www.brasilescola.com/fisica/o-que-sao-ondas-eletromagneticas.htm>. Acesso em 05 de abr.2013.

C2O: **Automação**. Disponível em <http://www.c2o.pro.br/automacao/x2045.html>. Acesso em 05 de abr. 2013.

CANALTECH: **Venda de smartphones**. Disponível em: <http://canaltech.com.br/noticia/smartphones/Venda-de-smartphones-Android-supera-iOS-no-segundo-trimestre/>. Acesso em 04 de abr.2013.

CHIPFIND: **Componentes Eletrônicos**. Disponível em: <http://www.chipfind.ru/catalog/sensors/objectsensor/h21a1.htm>. Acesso em 11 de jul.2013.

DARWIN, Ian F., **Android Cookbook**. São Paulo: Novatec, 2012.

DXBRASIL: **DX e Contest**. Disponível em: <http://www.dxbrasil.net/wp/artigo/mapa-dos-radioamadores/>. Acesso em 06 de mai.2013.

ENGEHERTZ: **Antenas**. Disponível em: <http://www.engehertz.com.br/#!Antena-Direcional-Yagi-Uda-/zoom/mainPage/image7fl>. Acesso em 25 de abr.2013.

GLOBALTOP: **Variação Magnética**. Disponível em: <http://www.gtop-tech.com/pt/product/Software_Services_16.html>. Acesso em 14 de abr.2013.

JAVAHISPANO: **Alguns Conceitos de Android**. Disponível em: <http://www.javahispano.org/android/2012/1/27/algunos-conceptos-de-android-antes-del-hola-mundo.html>. Acesso em 01 de mai.2013.

LABRE-PR: **Apostila para Ingresso ao Serviço de Radioamador**, 2010.

LECHETA, Ricardo R., **Google Android – Aprenda a criar aplicações para dispositivos móveis com o Android SDK**. São Paulo: Novatec, 2010.

LIANG, Daniel Y., **Introduction to Java Programming.** 8ª ed., United States of America: Prentice Hall, 2011.

McRoberts, Michael, **Arduino Básico**. São Paulo: Novatec, 2011.

NOT1: **Coordenadas Geográficas**. Disponível em: <http://www.not1.xpg.com.br/coordenadas-geograficas-latitude-e-longitude-fusos-horarios/>. Acesso em 11 de abr.2013.

PROJTEC: **Antenas e Propagação**. Disponível em: <http://www.projetostecnologicos.com/Cursos/Telecomunicacoes/Antenas/Antenas.html>. Acesso em 05 de abr.2013.

PROTEVE: **Ondas de Rádio**. Disponível em: <http://www.proteve.net/ondasderadio.html>. Acesso em 05 de abr.2013.

ROBOCORE: **Fórum Robocore**. Disponível em <http://www.robocore.net/modules.php?name=Forums&file=viewtopic&t=2364>. Acesso em 16 de abr.2013.

TELECO: **Redes sem Fio.** Disponível em: <http://www.teleco.com.br/tutoriais/tutorialbluezig/pagina_1.asp>. Acesso em 16 de abr.2013.

UFF: Universidade Federal Fluminense. **Tutorial sobre Motores de Passo**. Disponível em: <http://www.telecom.uff.br/pet/petws/downloads/tutoriais/stepmotor/stepmotor2k81119.pdf>. Acesso em 16 de abr.2013.

UFRGS: Universidade Federal do Rio Grande do Sul. **Astronomia de Posição**. Disponível em: <http://www.if.ufrgs.br/~fatima/fis2016/aulas/pratica_azimute.htm>. Acesso em 14 de abr.2013.

WITHUS**: Arduino Bluetooth Module Wireless Transceiver.** Disponível em: <http://withus.ru/node/77>. Acesso em 11 de jul.2013.

YAESU: **Rotators**. Disponível em: <http://www.yaesu.com/indexVS.cfm?cmd=DisplayProducts&ProdCatID=104&encProdID=60F24E075DF1D12D9CB5AD4A9C0A6855&DivisionID=65&isArchived=0>. Acesso em 14 de abr.2013.

6. ANEXO A

```
/*
UTFPR - UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
ESPECIALIZAÇÃO EM TECNOLOGIA JAVA E PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS
HIDERALDO LUIZ ZANETTI
06/2013
*** PROGRAMA PARA CONTROLE DE ANTENAS - PLATAFORMA ARDUINO ***
*/
#include <NewSoftSerial.h>

const int TOTAL_PASSOS = 96;
const int MOTOR_VELOCIDADE = 10;

const int antena1SensorPin = 2;
const int antena1antenaPin1 = 3;
const int antena1antenaPin2 = 4;
const int antena1antenaPin3 = 5;
const int antena1antenaPin4 = 6;

const int antena2SensorPin = 11;
const int antena2antenaPin1 = 7;
const int antena2antenaPin2 = 8;
const int antena2antenaPin3 = 9;
const int antena2antenaPin4 = 10;

int sensorPin = 0;
int antenaPin1 = 0;
int antenaPin2 = 0;
int antenaPin3 = 0;
int antenaPin4 = 0;

int posicaoAtual = 0;
int numeroPasso = 0;

int antena = 0;
int angulo = 0;
```

```
void setup() {

    Serial.begin(9600);

    selecionarAntena1();
    pinMode(sensorPin, INPUT);
    pinMode(antenaPin1, OUTPUT);
    pinMode(antenaPin2, OUTPUT);
    pinMode(antenaPin3, OUTPUT);
    pinMode(antenaPin4, OUTPUT);
    inicializarAntena();

    selecionarAntena2();
    pinMode(sensorPin, INPUT);
    pinMode(antenaPin1, OUTPUT);
    pinMode(antenaPin2, OUTPUT);
    pinMode(antenaPin3, OUTPUT);
    pinMode(antenaPin4, OUTPUT);
    inicializarAntena();
}

void loop() {
 if (Serial.available() > 0) {
     delay(100);
     char parametro = Serial.read();
     char codAntena[4];
     char posicao[4];

     codAntena[0] = Serial.read();
     codAntena[1] = Serial.read();
     codAntena[2] = Serial.read();
     codAntena[3] = '\0';

     posicao[0] = Serial.read();
     posicao[1] = Serial.read();
     posicao[2] = Serial.read();
     posicao[3] = '\0';

     if (parametro == 'P') {
```

```
        antena = atoi(codAntena);
        angulo = atoi(posicao);

        if (antena == 1) {
            selecionarAntena1();
        } else if (antena == 2) {
            selecionarAntena2();
        }

        if (angulo < 360) {
            posicionarAntena();
        } else if (angulo == 999) {
            inicializarAntena();
        }
      }
    }
 }

void posicionarAntena() {
    float passos = (TOTAL_PASSOS / 360.0) * angulo;
    int novaPosicao = (int)passos;
    if (posicaoAtual < novaPosicao) {
        passoFrente(novaPosicao - posicaoAtual, MOTOR_VELOCIDADE);
    } else if (posicaoAtual > novaPosicao) {
        passoAtras(posicaoAtual - novaPosicao, MOTOR_VELOCIDADE);
    }
    posicaoAtual = novaPosicao;
}

void inicializarAntena() {
  // LOW = POSIÇÃO ZERO - SENSOR ACIONADO
    if (digitalRead(sensorPin) == LOW) {
        passoFrente(24, MOTOR_VELOCIDADE);
    } else {
        for (int i = 0;i < 24;i++) {
            passoFrente(1, MOTOR_VELOCIDADE);
            if (digitalRead(sensorPin) == LOW) {
                while (digitalRead(sensorPin) == LOW) {
                    passoAtras(1, MOTOR_VELOCIDADE);
```

```
                }
                break;
            }
        }
    }
    delay(100);
    while (digitalRead(sensorPin) == HIGH) {
        passoAtras(1, MOTOR_VELOCIDADE);
    }
    posicaoAtual = 0;
    delay(100);
    pararAntena();
}

void selecionarAntena1() {
    sensorPin = antena1SensorPin;
    antenaPin1 = antena1antenaPin1;
    antenaPin2 = antena1antenaPin2;
    antenaPin3 = antena1antenaPin3;
    antenaPin4 = antena1antenaPin4;
}

void selecionarAntena2() {
    sensorPin = antena2SensorPin;
    antenaPin1 = antena2antenaPin1;
    antenaPin2 = antena2antenaPin2;
    antenaPin3 = antena2antenaPin3;
    antenaPin4 = antena2antenaPin4;
}

void passoFrente(int passos, int rpm) {
    int esperaMs = (60000 / (TOTAL_PASSOS * rpm));
    for (int i = 0;i < passos;i++) {
        passoAntena(1);
        delay(esperaMs);
    }
    pararAntena();
}
```

```
void passoAtras(int passos, int rpm) {
    int esperaMs = (60000 / (TOTAL_PASSOS * rpm));
    for (int i = passos;i > 0;i--) {
        passoAntena(-1);
        delay(esperaMs);
    }
    pararAntena();
}

void passoAntena(int direcao) {
  switch (numeroPasso) {
      case 0:
          digitalWrite(antenaPin1, HIGH);
          digitalWrite(antenaPin2, LOW);
          digitalWrite(antenaPin3, LOW);
          digitalWrite(antenaPin4, LOW);
          break;
     case 1:
          digitalWrite(antenaPin1, HIGH);
          digitalWrite(antenaPin2, HIGH);
          digitalWrite(antenaPin3, LOW);
          digitalWrite(antenaPin4, LOW);
          break;
    case 2:
          digitalWrite(antenaPin1, LOW);
          digitalWrite(antenaPin2, HIGH);
          digitalWrite(antenaPin3, LOW);
          digitalWrite(antenaPin4, LOW);
          break;
    case 3:
          digitalWrite(antenaPin1, LOW);
          digitalWrite(antenaPin2, HIGH);
          digitalWrite(antenaPin3, HIGH);
          digitalWrite(antenaPin4, LOW);
          break;
    case 4:
          digitalWrite(antenaPin1, LOW);
          digitalWrite(antenaPin2, LOW);
          digitalWrite(antenaPin3, HIGH);
```

```
            digitalWrite(antenaPin4, LOW);
            break;
        case 5:
            digitalWrite(antenaPin1, LOW);
            digitalWrite(antenaPin2, LOW);
            digitalWrite(antenaPin3, HIGH);
            digitalWrite(antenaPin4, HIGH);
            break;
        case 6:
            digitalWrite(antenaPin1, LOW);
            digitalWrite(antenaPin2, LOW);
            digitalWrite(antenaPin3, LOW);
            digitalWrite(antenaPin4, HIGH);
        case 7:
            digitalWrite(antenaPin1, HIGH);
            digitalWrite(antenaPin2, LOW);
            digitalWrite(antenaPin3, LOW);
            digitalWrite(antenaPin4, HIGH);
    }
    if (direcao == 1) {
        numeroPasso++;
        if (numeroPasso > 7) {
            numeroPasso = 0;
        }
    } else {
        numeroPasso--;
        if (numeroPasso < 0) {
            numeroPasso = 7;
        }
    }
}

void pararAntena() {
    digitalWrite(antenaPin1, LOW);
    digitalWrite(antenaPin2, LOW);
    digitalWrite(antenaPin3, LOW);
    digitalWrite(antenaPin4, LOW);
}
```

7. ANEXO B

```
package infocamp.net.btac;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.Set;
import java.util.UUID;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;

public class Controle {

	private ArrayList<Antena> antenas = new ArrayList<Antena>();
	private Antena antenaSelecionada;
	private Mensagem mensagem = new Mensagem();
	byte[] buffer = new byte[1024];

	private BluetoothDevice dispositivo;
	private BluetoothSocket socket = null;

	private static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-
8000-00805f9b34fb");

	public Mensagem bluetoothInicializar(String nomeDispositivo) {

		BluetoothAdapter adapter = BluetoothAdapter.getDefaultAdapter();
		if (adapter==null) {
			mensagem.setError(true);
			mensagem.setCodigo(R.string.btadaptadornaoencontrado);
		} else if (!adapter.isEnabled()) {
			mensagem.setError(true);
			mensagem.setCodigo(R.string.btadaptadordesabilitado);
		} else {
			Set<BluetoothDevice> devices = adapter.getBondedDevices();
			dispositivo = null;
			for (BluetoothDevice device : devices) {
				if (nomeDispositivo.equals(device.getName())) {
					dispositivo = device;
				}
			}
			if (dispositivo==null) {
				mensagem.setError(true);
				mensagem.setCodigo(R.string.btnaopareado);
			} else {
				mensagem.setError(false);
				mensagem.setCodigo(0);
				bluetoothConectar();
			}
		}
		return(mensagem);
	}
```

```
	private Mensagem bluetoothConectar() {
		Thread conector = new Thread(new Runnable() {
			@Override
			public void run() {
				UUID ident = MY_UUID;
				try {
					socket =
dispositivo.createRfcommSocketToServiceRecord(ident);
					socket.connect();
					mensagem.setError(false);
					mensagem.setCodigo(0);
					bluetoothReceber();
				} catch (IOException e) {
					mensagem.setError(true);
					mensagem.setCodigo(R.string.erroconexao);
				}
			}
		});
		conector.start();
		try {
			conector.join();
		} catch (InterruptedException e) {
		}
		return(mensagem);
	}

	public Mensagem criarAntena(int codigoAntena) {
		Antena antena = new Antena();
		antena.setCodigo(codigoAntena);
		antena.setPosicao(0);
		antenas.add(antena);
		mensagem.setError(false);
		mensagem.setCodigo(0);
		return(mensagem);
	}

	public Antena getAntenaSelecionada() {
		return antenaSelecionada;
	}

	public Mensagem selecionarAntena(int codigoAntena) {
		for (Antena antena : antenas) {
			if (antena.getCodigo()==codigoAntena) {
				antenaSelecionada = antena;
			}
		}
		mensagem.setError(false);
		mensagem.setCodigo(0);
		return(mensagem);
	}

	public Mensagem inicializarAntena() {
		String codigo = ("000" + antenaSelecionada.getCodigo());
		codigo = codigo.substring(codigo.length()-3, codigo.length());
		String comando = "P" + codigo + "999";
		bluetoothEnviar(comando);
		posicionarAntena(0);
		mensagem.setError(false);
		mensagem.setCodigo(0);
```

```
		return(mensagem);
	}

	public Mensagem posicionarAntena(int posicaoAntena) {
		String codigo = "000" + antenaSelecionada.getCodigo();
		codigo = codigo.substring(codigo.length()-3, codigo.length());
		String posicao = "000" + posicaoAntena;
		posicao = posicao.substring(posicao.length()-3, posicao.length());
		String comando = "P" + codigo + posicao;
		bluetoothEnviar(comando);
		mensagem.setError(false);
		mensagem.setCodigo(0);
		return(mensagem);
	}

	private void bluetoothReceber() {
		Thread thread = new Thread() {
			@Override
			public void run() {
				try {
					InputStream input = socket.getInputStream();
					input.read(buffer);
				} catch (IOException e) {
				}
			}
		};
		thread.start();
	}

	private Mensagem bluetoothEnviar(final String comando) {
		Thread thread = new Thread() {
			@Override
			public void run() {
				try {
					OutputStream output = socket.getOutputStream();
					output.write(comando.getBytes());
					mensagem.setError(false);
					mensagem.setCodigo(0);
				} catch (IOException e) {
					mensagem.setError(true);
					mensagem.setCodigo(R.string.bterroenvio);
				}
			}
		};
		thread.start();
		return(mensagem);
	}

	public Mensagem bluetoothEncerrar() {
		try {
			socket.close();
		} catch (IOException e) {
			mensagem.setError(true);
			mensagem.setCodigo(R.string.bterroconexao);
		}
		return(mensagem);
	}
}
```

```java
package infocamp.net.btac;

public class Antena {
	private int codigo;
	private int posicao;

	public int getCodigo() {
		return codigo;
	}

	public void setCodigo(int codigo) {
		this.codigo = codigo;
	}

	public int getPosicao() {
		return posicao;
	}

	public void setPosicao(int posicao) {
		this.posicao = posicao;
	}
}
```

```java
package infocamp.net.btac;

public class Mensagem {
	private boolean error;
	private int codigo;

	public boolean hasError() {
		return error;
	}
	public void setError(boolean error) {
		this.error = error;
	}
	public int getCodigo() {
		return codigo;
	}
	public void setCodigo(int codigo) {
		this.codigo = codigo;
	}
}
```

8. ANEXO C

Lista de Materiais

Quantidade	Material
1	Arduino Uno
1	Shield Bluetooth
2	Sensor fotoelétrico H21A1
2	Circuito Integrado ULN2003
2	Motor de Passo
2	Resistor 330 Ω
2	Resistor 2,2 KΩ
2	Diodo Zenner 12 V 0,5 W