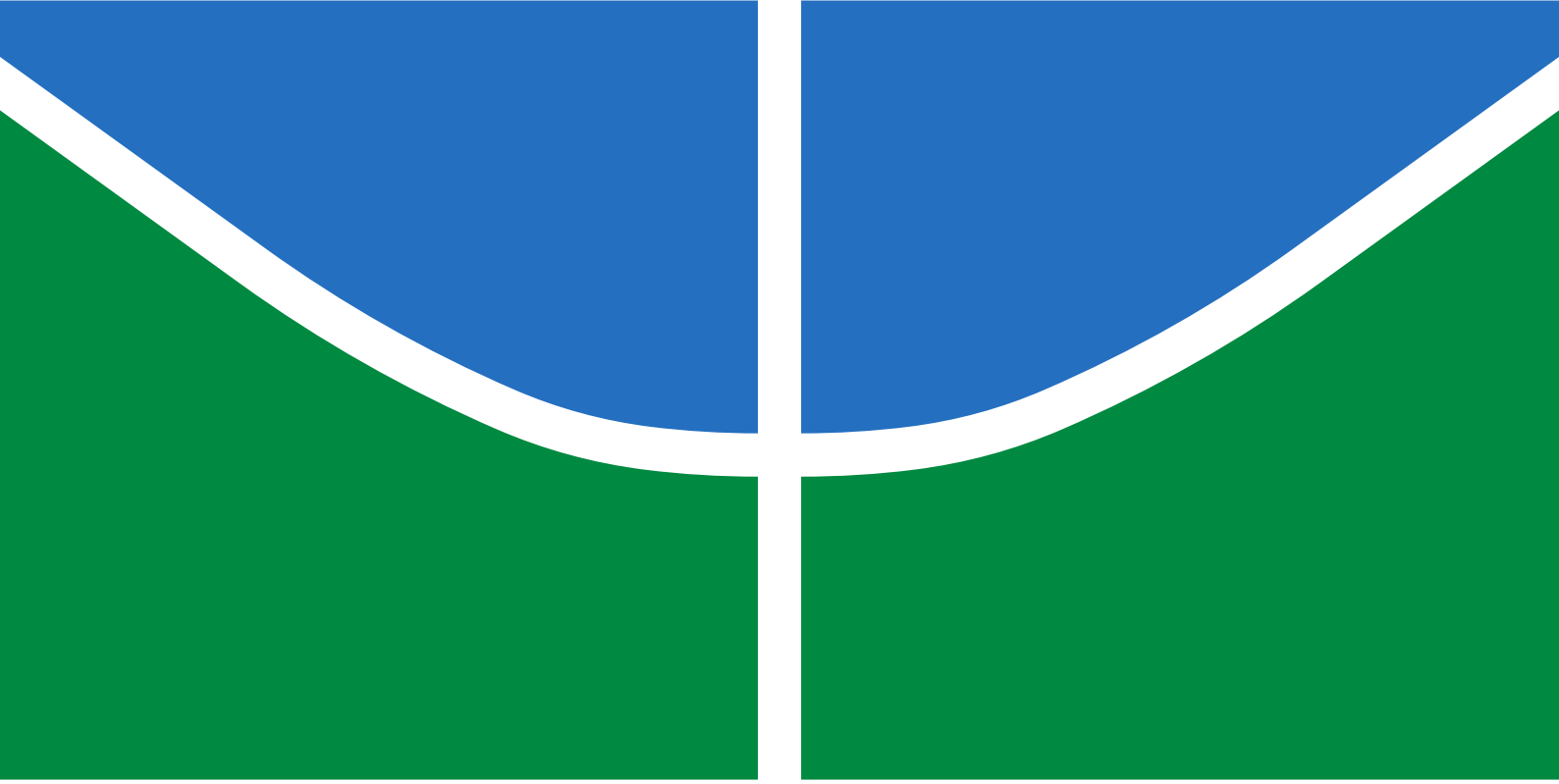


TRABALHO DE GRADUAÇÃO

Sistema de Segurança e Localização de Objetos em Ambientes Prediais e Residências

Érik Yuiti Ohara

Brasília, Abril de 2013

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

Sistema de Segurança e Localização de Objetos em Ambientes Prediais e Residências

Érik Yuiti Ohara

Relatório submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro Eletricista

Banca Examinadora

Prof. Adolfo Bauchspiess, ENE/UnB _____
Orientador

Prof. Lélio Ribeiro Soares Júnior, ENE/UnB _____
Examinador interno

Prof. Franklin da Costa Silva, ENE/UnB _____
Examinador interno

Agradecimentos

Gostaria de agradecer primeiramente ao meu pai e a minha mãe. Nunca poderei agradecer o bastante por tudo que eles fizeram por mim. Toda a educação, a atenção, o carinho e o amor. Tudo o que sou, devo a eles. Meus avôs, minhas irmãs, meus primos e toda minha família sempre foi um enorme suporte para todos os momentos da minha vida.
Quero agradecer aos meus amigos da engenharia elétrica. Nos estudos, nas festas, nos maus e bons momentos estávamos sempre juntos. Amizade é essencial e eu não teria conseguido passar por esses 5 anos se não fossem as amizades.
Agradeço aos colegas Frederico Rocha e Filipe Oliveira, que me ajudaram muito no desenvolvimento desse trabalho.
E aos professores. Professor é uma profissão que eu admiro muito e sou muito agradecido por todos os ensinamentos que me passaram durante essa graduação. Também agradeço a todos os funcionários do departamento de engenharia elétrica e a Deus por sempre me guiar pelos caminhos que levam à realização dos meus sonhos.

Érik Yuiti Ohara

RESUMO

Nesse trabalho, foi desenvolvido um sistema para realizar a localização, a segurança e o controle dos objetos em ambientes prediais e residenciais. Para isso, foram utilizada a tecnologia RFID (Radio Frequency IDentification). Utilizou-se a linguagem de programação Java para o desenvolvimento dos softwares utilizados nesse projeto. O sistema foi implantando e simulado no Laboratório de Automação e Robótica da UnB com o objetivo de analisar a viabilidade de ser aplicado em residências.

Palavras Chave: RFID, antenas, localização, segurança, objetos, Java, Android

ABSTRACT

In this paper, a system was developed to perform the location, security and control of objects in building environments and residential. For this, it was used the RFID (Radio Frequency IDentification).The Java programming language was used for the development of software used in this project. The system was implemented and simulated in the Laboratory of Automation and Robotics UNB in order to analyze the feasibility to be applied in homes.

Keywords: RFID, antennas, location, security, objects, Java, Android

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1

Introdução

1.1 Contextualização

Onde está minha chave? Alguém viu o documento do carro? Cadê o livro que eu deixei aqui? Será que eu deixei minha carteira em casa? À medida em que se intensifica o volume de obrigações diárias, essas perguntas se tonam cada vez mais comuns. Devido a competição no mercado de trabalho, a preocupação com o emprego acaba sendo prioridade na mente das pessoas, que acabam esquecendo de assuntos menores como a localização de seus pertences.

Uma outra consequência, é a terceirização de serviços de casa como: limpeza, manutenção de aparelhos elétricos, desentupimento de encanamento. Tais problemas eram comumente resolvidos pelos próprios proprietários da casa, mas, para muitos, é mais viável e menos estressante contratar um profissional do quer perder tempo resolvendo por si mesmo. Um grande problema nesse processo está na confiança com o trabalhador. É seguro deixa-lo sozinho na casa com todos os pertences desprotegidos?

1.2 Objetivos do projeto

Esse projeto visa resolver esses problemas através de um sistema de localização de objetos através da tecnologia RFID, acoplada em um software SCADA Action View.

• Comunicação entre o supervisório e o módulo de leitura de dados de etiquetas RFID e do supervisório e o celular;

• Desenvolvimento de aplicativo de celular para controle e planejamento de fluxo dos objetos;

• Disponibilização de histórico de passagem dos objetos pelas antenas para localização dos mesmos.

O projeto também visa mostrar a viabilidade da aplicação do sistema em uma residência.

1.3 Caracterização do Projeto

O projeto consiste em um sistema de controle, localização e segurança de objetos em uma residência. Primeiramente, todos os objetos selecionados irão possuir uma etiqueta RFID. Dessa forma, cada um terá um ID especificado. Então, essas etiquetas são lidas por antenas que passam a informação para um supervisório. Esse grava a informação do ID e da antena que realizou a leitura. Haverão antenas na entrada de cada cômodo da residência. Dessa forma, será possível ao supervisório definir a localização do objeto. O supervisório então passa o ID para um celular. Esse possui um aplicativo onde o usuário define o nome do objeto, o fluxo de saída da residência do mesmo e as opções de alerta. Ele pode definir que dias da semana sai com o objeto ou algum dia específico. Depois, define se quer um alerta caso saia de sua residência sem o objeto nos dias especificados. Também pode colocar um alerta em caso do objeto sair sem estar em sua posse.

Assim, o usuário poderá controlar o fluxo de saída do objeto de sua residência; localizar em qual cômodo ele se encontra; e receber notificações em caso de saída do objeto desautorizada.

1.4 Apresentação do manuscrito

Este manuscrito encontra-se organizado da seguinte forma: no capítulo 2 é feita uma fundamentação teórica sobre java, android, RFID e antenas. No capítulo 3, é feita uma descrição sobre o desenvolvimento do aplicativo de celular e do programa supervisório. No capítulo 4, são apresentadas as simulações, seus resultados e as análises. Também é analisado a viabilidade de se aplicar o projeto em uma residência. No capítulo 5 encontra-se a conclusão. Finalmente nos anexos encontram-se os códigos desenvolvidos, planejamento das simulações, histórico do programa resultante da simulação 2 e cotação de seguro.

Capítulo 2

Conceitos e Ferramentas Utilizadas no Projeto

2.1 Java

A linguagem de programação Java, foi utilizada por ser a linguagem comumente utilizada para desenvolvimento de aplicativos para celulares. Desenvolvida pela SunMicrosystems, ela é uma linguagem com código aberto que é compilada em bytecodes, interpretados pelas Máquinas Virtuais Java (JVM – Java Virtual Machine). Desse modo, os programas Java são independentes de plataforma (figura 2.1) [8]. A arquitetura java é definida por:

- A linguagem de programação Java
- O formato do arquivo .class
- A máquina vitual Java
- A API(Application Programming Interface) Java

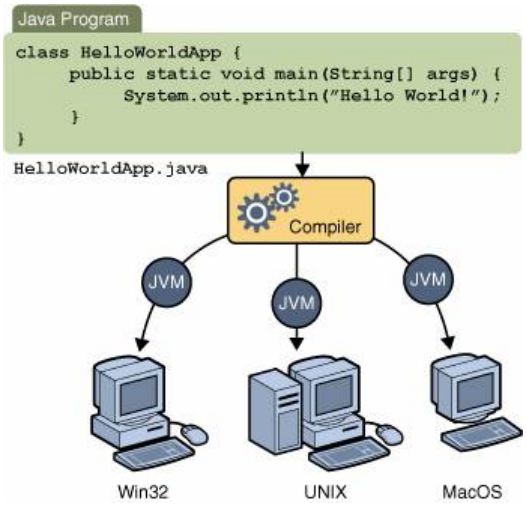


Figura 2.1: Portabilidade da liguagem Java.

A linguagem Java é interpretada e compilada ao mesmo tempo, ou seja, os bytecodes são gerados pela máquina virtual e interpretados no computador simultaneamente(figura2.2). O interpretador ainda fornece um ambiente de execução, com garbage collector para o gerenciamento de memória e a funcionalidade necessária para acessar arquivos .class.

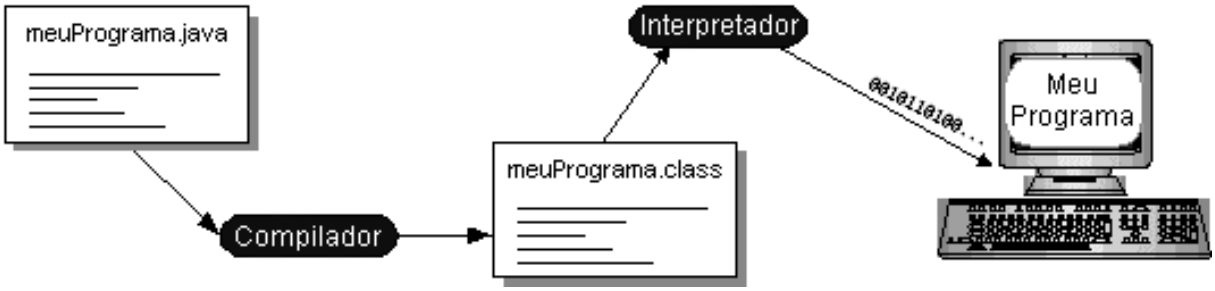


Figura 2.2: Compilador e Interpretador Java

Java é uma linguagem orientada a objeto. Isso significa que são implementados um conjunto de classes que definem objetos presentes no sistema de software. Cada classe determina o comportamento (definido nos métodos) e estados possíveis (atributos) de seus objetos, assim como o relacionamento com outros objetos. Assim, já existem várias bibliotecas com classes e interfaces que facilitam a programação.[9]

2.2 Eclipse

Eclipse é um IDE (Integrated Development Environment), um ambiente integrado para desenvolvimento de software. Ou seja, é um programa editor utilizado para facilitar a geração do código e futuras correções [1]. O processo de desenvolvimento é separado em 5 etapas:

2.2.1 Fase 1 - Editor

É a fase onde o arquivo .java é criado e editato. Normalmente realizado em uma IDE, esse arquivo contém o programa a ser executado.

Figura 2.3: Programação Java - Etapa 1

2.2.2 Fase 2 - Compilador

O programa é traduzido para bytecodes, ou linguagem de máquina caso não haja falhas no código. São gerados então arquivos do tipo .class, contendo os bytecodes que serão utilizados

durante a execução do programa.

Figura 2.4: Programação Java - Etapa 2

2.2.3 Fase 3 - Carregador de classe

Nessa fase, o programa é armazenado na memória para executá-lo, efetuando o carregamento. O carregador pega os arquivos .class que contém os bytecodes do programa e transfere para a memória primária. Existe a possibilidade também de serem transferidos para um computador remoto, que usuários conectam via Internet (applets - uso industrial). Para acessar um applet novamente, o usuário deve apontar seu navegador para a localização na Internet e carregar o programa no navegador. Para acessar um aplicativo, o usuário pode simplesmente cliquar sobre o ícone e executá-lo, se estiver utilizando sistemas operacionais como Windows ou Linux

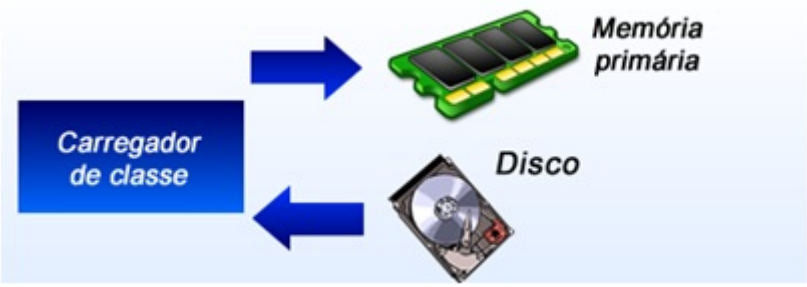


Figura 2.5: Programação Java - Etapa 3

2.2.4 Fase 4 - Verificador de bytecode

Enquanto as classes são carregadas, o verificador examina seus bytecodes para assegurar que são válidos e não violam restrições de segurança do Java. Essa funcionalidade faz com que o Java seja uma linguagem segura, pois impede que pro- gramas descarregados pela rede causem danos aos seus arquivos e seu sistema.

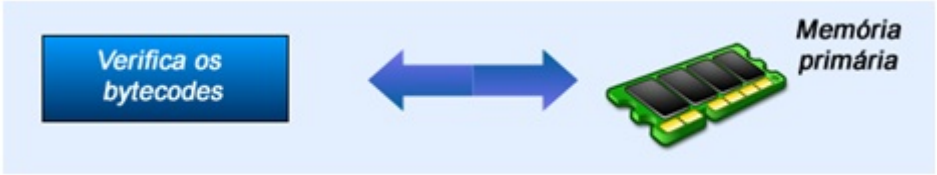


Figura 2.6: Programação Java - Etapa 4

2.2.5 Fase 5 - Máquina Virtual Java

As JVMs executam bytecodes utilizando uma combinação de interpretação chamada compilação JIT (Just In Time) - conhecido como compilador Java HotSpot.

Figura 2.7: Programação Java - Etapa 5

Através do Eclipse (figura 2.8), é possível realizar todas as tarefas descritas acima. Quando o programador salva seu código localmente, as funções compilação, carga e verificação são realizadas automaticamente pelo editor. Nele, também é possível realizar a execução do programa. Dessa forma, já é possível identificar falhas no processo de criação do código e erros de lógica ou sintaxe, sem necessidade de finalização do código para futuros testes e identificação de erros.

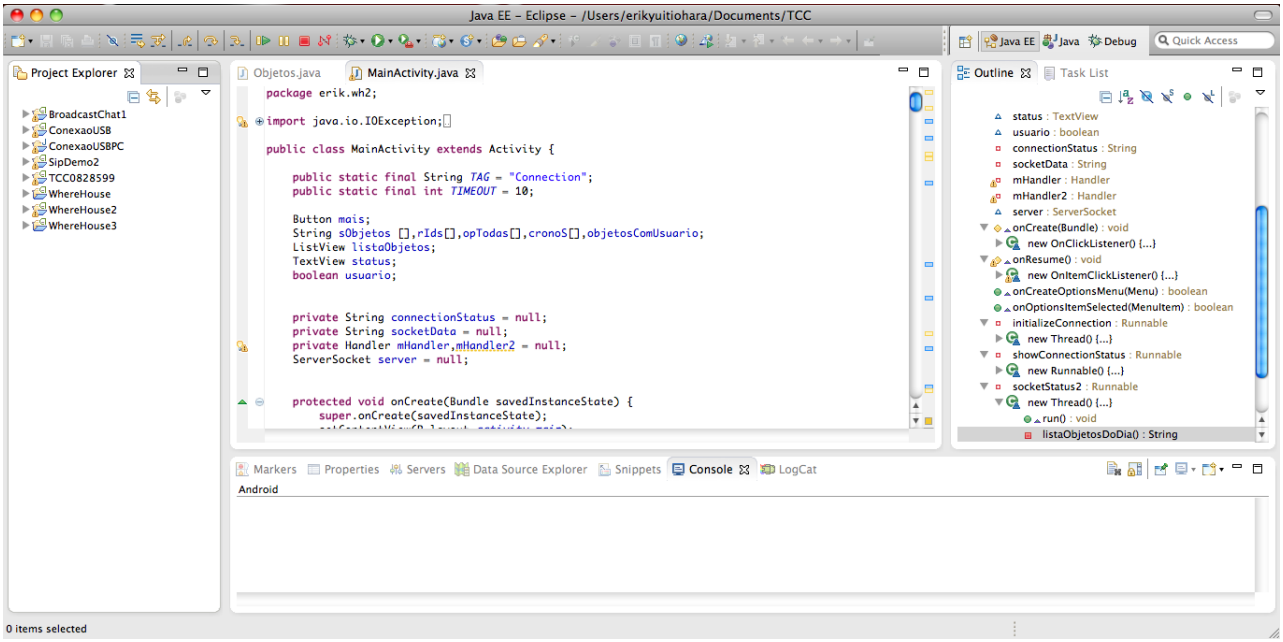


Figura 2.8: Eclipse

2.3 Android

Android é atualmente o maior sistema operacional presente em plataforma de aparelhos móveis (figura 2.9). Ele já oferece inúmeras bibliotecas para facilitar o desenvolvimento de aplicativos para celulares. Esses são escritos na linguagem de programação JAVA. O android já oferece um SDK (Software Development Kit), ou seja, Kit de Desenvolvimento de Software. Através dele, o programa editado no Eclipse, por exemplo, é compilado em um pacote Android, no formato .apk.

Todo arquivo .apk representa um aplicativo e é este que os aparelhos celulares usam para instalar os aplicativos. [2]

Figura 2.9: Evolução da plataforma Android

Os aplicativos são divididos em 4 componentes: atividades, serviços, provedores de conteúdo e receptores de Radiodifusão. Desse 4, utilizaremos apenas a atividade.

2.3.1 Atividade

Uma atividade é uma única tela de interface com o usuário. Através dela, cada usuário pode interagir com algum objetivo específico como: ligar para um número, tirar uma foto, enviar um email.

Um aplicativo geralmente possui várias atividades, relacionadas umas com as outras. Em geral, existe uma atividade principal, apresentada quando o aplicativo é iniciado pela primeira vez, e a partir dela, várias outras atividades são iniciadas com o objetivo de realizar alguma ação. Todo momento em que uma nova atividade é iniciada, a atividade anterior é parada, mas o sistema preserva a atividade em uma pilha (chamada "back stack"). Quando o usuário decide encerrar uma atividade, apertando o botão de voltar, ele destrói a atividade, fazendo com que o sistema puxe a atividade anterior da pilha, através da regra "último a entrar, primeiro a sair".

Assim, cada atividade segue o seguinte ciclo de vida:

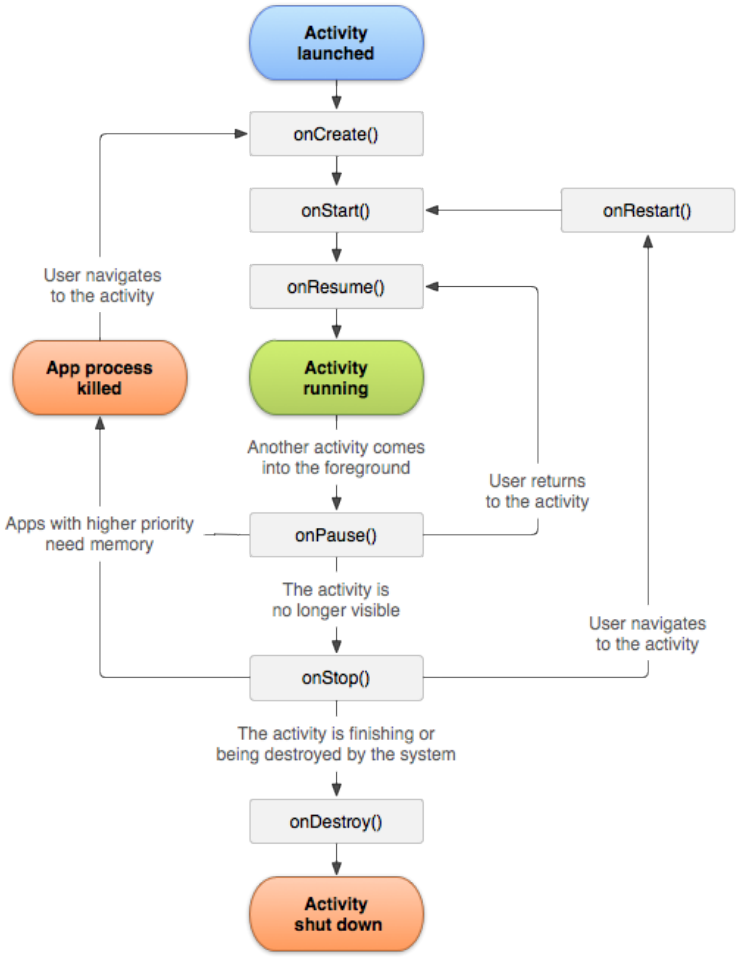


Figura 2.10: Ciclo de Vida de uma Atividade

As fases mais importantes da vida de uma aplicativo (as únicas utilizadas nesse trabalho) são OnCreate() e OnResume(). Em OnCreate(), o programador deve colocar todas as informações iniciais de uma atividade: como iniciar variáveis, relacionar variáveis com a interface gráfica. E em OnResume(), devem ser postas as informações que devem ser atualizadas quando a atividade é retomada como: alterações de variáveis, ou de configurações.

Toda atividade possui uma ação. É através dela que as atividades são iniciadas. Pela intenção de uma ação, é possível iniciar uma atividade de até outro aplicativo.

2.3.2 Manifesto

Para criar uma atividade, é necessário declará-la no manifesto. O manifesto é um arquivo presente em todo o aplicativo onde são colocadas todas as informações essenciais sobre o aplicativo para o sistema Android. Todas as informações que o sistema precisa saber antes de executar o aplicativo. Geralmente são elas:

- Nome do Pacote Java

O nome do pacote é único para cada aplicativo.

• Componentes do aplicativo

Todas as atividades, serviços, provedores de Conteúdo e Receptores de Radiodifusão. Também são informadas suas respectivas ações e classes Java.

• Permissões

Contém tudo que o aplicativo necessita acessar do aparelho celular para o seu funcionamento. Por exemplo: acessar a Internet, acessar a câmara fotográfica, enviar SMS, utilizar GPS.

• Requisitos básicos do Sistema

Onde está a versão mínima do sistema operacional para o funcionamento do aplicativo. Lista de bibliotecas utilizadas.

2.3.3 Interface Gráfica e Recursos

Figura 2.11: Interface Gráfica do Android

A interface gráfica é tudo que o usuário pode ver e interagir. O Android oferece uma diversidade de componentes e layouts para permitir ao programador desenvolver o seu aplicativo. Geralmente, toda atividade possui um layout em formato .xml, onde são estruturados todos os componentes e suas posições na tela. Tais arquivos localizam-se na pasta layout, encontrada dentro da pasta res(recursos). Na pasta de recursos, também são encontradas arquivos de imagens presentes no aplicativo, layout dos menus e também variáveis com valores constantes.

2.4 RFID

RFID (Radio Frequency Identification), ou identificação por radio frequência, é um termo para tecnologias que utilizam ondas de rádio para identificar objetos ou pessoas. O método mais comum é armazenar um número serial que identifica uma pessoa ou objeto em um microchip ligado a uma antena. A antena transmite as informações do chip para uma leitora, que converte as ondas de

rádio em informações digitais que possam ser utilizadas por computadores. [6]

Um sistema RFID comporta três componentes: etiqueta, antenas e a leitora. A etiqueta trata-se da junção de seu chip e sua antena acoplada citados anteriormente. As antenas (que são de recepção) podem ser acopladas com a leitora ou ligadas com elas por um cabo. A leitora pode ter uma interface própria ou, mais comumente, estar ligada a um computador da onde o usuário irá acessar, armazenar e controlar os dados resultantes.

2.4.1 Etiquetas

Etiquetas são cominações de um microchip com uma antena utilizados para identificar objetos ou pessoas. Existem diversos tipos, definidos dependendo do objeto (valor e tamanho), do ambiente, da distância da etiqueta e da leitora, dentre outros fatores. Existem algumas etiquetas que possuem também uma pequena bateria, e algumas até sensores. O chip possui memória que pode ser lida, mas em alguns casos, memória que pode ser escrita. Além disso, em alguns casos mais raros, até processamento de dados.

Geralmente, classificamos as etiquetas de três formas:

- Classificação segundo a alimentação;
- Classificação segundo a memória;
- Classificação EPC Global;

2.4.1.1 Classificação segundo a alimentação

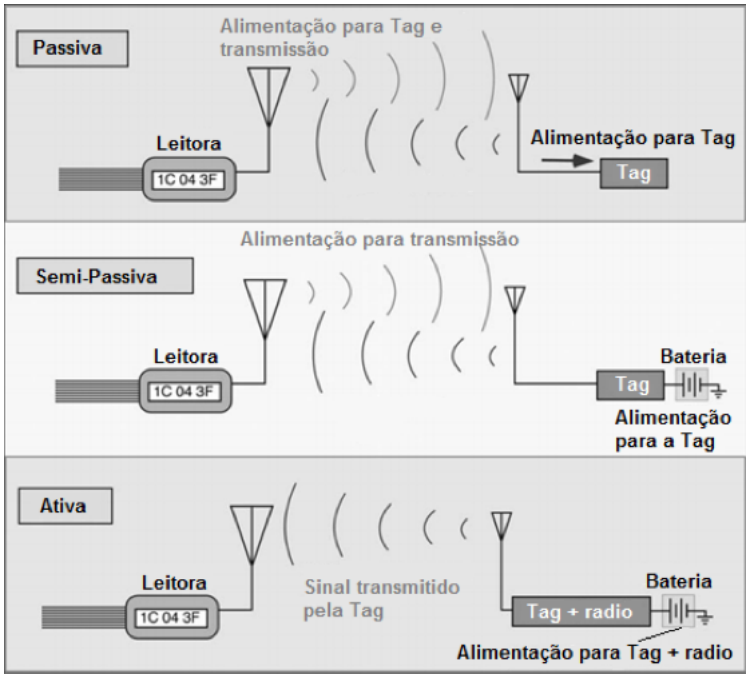


Figura 2.12: Classificação das etiquetas em relação a alimentação

Pela alimentação, as etiquetas RFID possuem três tipos:

- Etiquetas Ativas;
- Etiquetas Semi-passivas;
- Etiquetas Passivas. [4]

Etiquetas Ativas Etiquetas Ativas se diferenciam por possuir uma fonte interna de energia. São geralmente mais caras, mas por possuírem energia própria, podem ser detectadas em distâncias maiores da leitora. Também, é mais usual para esse tipo de etiqueta, a presença de sensores ou microprocessadores. Desse modo, podem ser utilizados para medir temperatura, calcular prazos de validade, e enviar tais dados para a leitora.

Algumas etiquetas permanecem em estado de dormência e só são ativadas, quando recebem algum sinal da leitora, economizando assim sua bateria. São chamadas Active Transponders. Outras enviam sinais a cada intervalo específico de tempo. Sendo muito utilizados para sistemas de localização, pois através do tempo de resposta, algumas leitoras possam triangular a posição da etiqueta. Essas são chamadas de Beacons.

Assim, tais etiquetas são recomendadas para objetos de valor relativamente grande e para detecção de maiores distâncias.

Etiquetas Semi-passivas Essas também possuem uma fonte de energia interna como as etiquetas ativas. Porém, o que diferencia é que a fonte alimenta apenas os circuitos internos . Assim, a transmissão do sinal utiliza a energia das ondas recebidas da leitora.

O custo e o alcance são intermediários em relação as etiquetas ativas e as passivas.

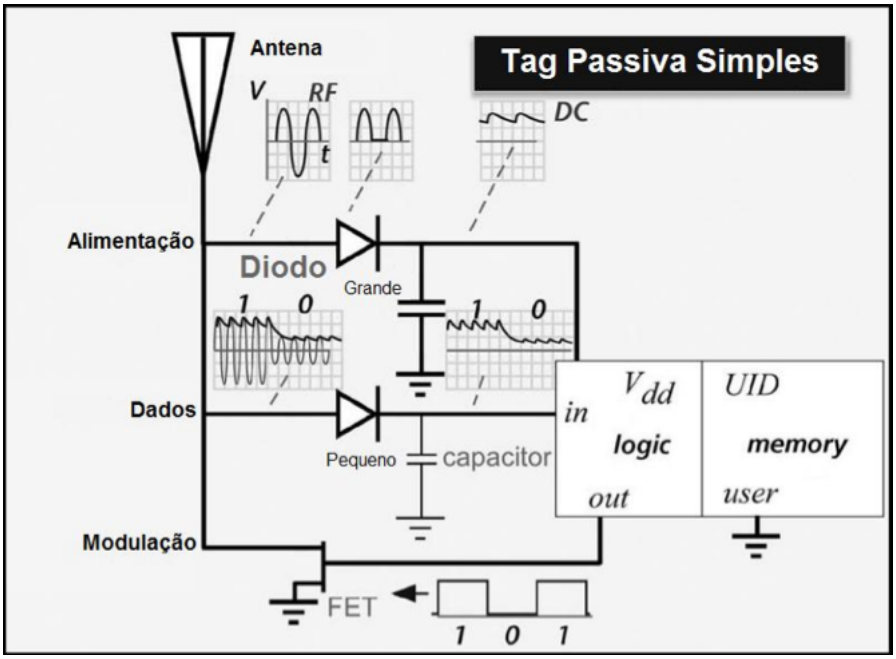


Figura 2.13: Etiqueta Passiva Simples

Etiquetas Passivas As passivas possuem apenas os componentes básicos citados anteriormente. Não possuem nenhuma fonte de energia, o que diminui seu custo consideravelmente. Utiliza as ondas recebidas da leitora para alimentar os seus circuitos internos e para enviar um sinal de resposta.

Ao receber o sinal da leitora, a antena converte o campo eletromagnético em corrente através de suas espiras. (como mostra a figura 2.13). Tal corrente é retificada pelo conjunto diodo-capacitor na área de alimentação e também na área de dados. Estas duas áreas se diferem pelo tamanho do capacitor, que é grande na área alimentação para segurar a tensão em alta para alimentar os circuitos de logica/memoria da etiqueta e pequeno na área de dados para segurar a tensão alta apenas na ocorrência do bit 1 deixando a tensão baixa na ocorrência do 0.

2.4.1.2 Classificação segundo a memória

Segundo a memória, as etiquetas podem ser classificadas de duas formas:

- Etiquetas de 1-bit
- Etiquetas de n-bits

Essas duas categorias se subdividem como mostra a figura 2.14 [5]:

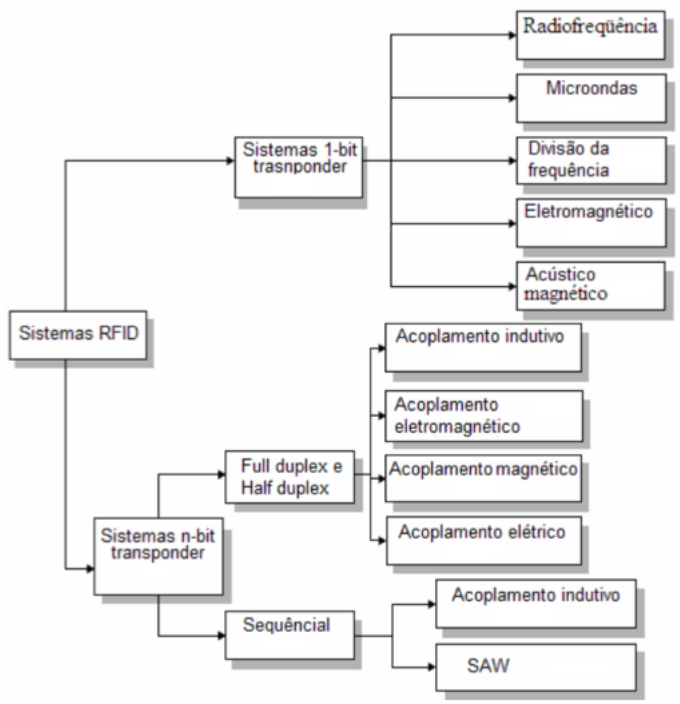


Figura 2.14: Classificação de etiquetas em relação a memória

Etiquetas de 1-bit Um bit é a menor unidade digital que pode assumir apenas dois valores: 0 ou 1. Tais valores significam apenas se a etiqueta está sendo lida pela leitora ou não está ao alcance dela. Apenas com essa informações, inúmeras práticas podem ser realizadas, incluindo

a desse presente trabalho. Geralmente, tratam-se de etiquetas passivas, pois é necessária uma quantidade muito grande de energia.[5]

Etiquetas de n-bit No caso de etiquetas de n-bits, há transmissão de dados entre a etiqueta e a leitora. Como possuem mais informações, geralmente, são etiquetas ativas ou etiquetas semi-passivas. Podem atuar de modo full-duplex, half-duplex e sequencial.

No modo full-duplex (FDX) a informação é transmitida simultaneamente da etiqueta para a leitora (uplink) e da leitora para a etiqueta (downlink), isto pode ser conseguido fazendo com que a etiqueta transmita em um sub- harmônico da frequência de transmissão da leitora ou mesmo em uma frequência totalmente não relacionada. Neste modo podem-se transmitir mais informações em um mesmo período de tempo do que os outros modos considerando uma mesma taxa de transmissão de dados.

Em half-duplex (HDX) a transmissão de informação apenas ocorre ou da leitora para a etiqueta (downlink) ou da etiqueta para a leitora (uplink), nunca se dá ao mesmo tempo.

Em modo sequencial (SEQ) a transmissão de energia da leitora para a etiqueta não é continuo como nos modos acima, neste caso a energia é transmitida por um período junto com a informação da leitora para a etiqueta, em pulsos. A transmissão de dados da etiqueta para a leitora ocorre no intervalo entre esses pulsos.

2.4.1.3 Classificação EPC Global

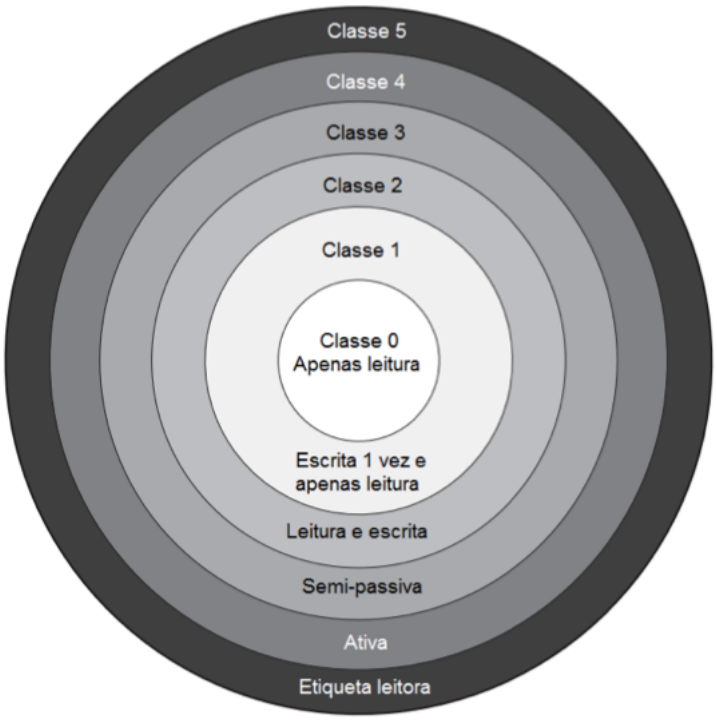


Figura 2.15: Classificação de etiquetas em relação a EPC global

Como já dito anteriormente, RFID utiliza ondas de rádio para relacionar objetos ou pessoas a números de identificação. Esses número se chamam EPC (Electronic Product Code), ou Código Eletrônico de Produtos e as etiquetas que o possuem são chamadas etiquetas EPC. A complexidade dos dados EPC dependem do número de funcionalidades da etiqueta. Quando mais forem, maior o custo, pois se faze necessário mais circuitos ou até uma bateria. Para de acomodar os inúmeros níveis de complexidade foram propostas 6 classes para as etiquetas como pode ser visto na figura 2.15. [7]

Classe 0 Etiquetas classe 0 são programadas com um único EPC durante sua fabricação, não podendo ser alterado pelo usuário. Por isso, são etiquetas simples e passivas, que podem apenas serem lidas. São apropriadas para situações em que se deseja apenas detectar a presença da mesma como dispositivos antirroubos. Uma etiqueta classe 0 deve ter as seguintes características:

-Numero identificador TID (The Tag Identifier). Que é único e atribuído pelo fabricante.

-Numero EPC . Também atribuído pelo fabricante para identificar o objeto no qual a etiqueta esta afixada.

-A função kill. Usada para desabilitar a etiqueta permanentemente.

Classe 1 Semelhante a classe 0, mas com a diferença de que pode ser escrita uma vez, devido ao seu acoplamento backscatter. Assim, ela possui as seguintes características:

-Passiva. Não podem iniciar a comunicação nem podem possuir fonte interna de energia.

-Memoria: Ter 128 bits de memoria, sendo que 96 bits destes são para armazenamento de sua identificação e 32 bits para correção de erro e a função kill.

-Memoria Write-once, read-many (WORM). Pode ser programada pelo fabricante ou pelo usuário, mas apenas uma vez.

Classe 2 Etiquetas classe 2 possuem mais memória do que apenas para identificação, pois são tipicamente utilizadas para registrar dados. Possuem as seguinte características:

-Passiva. Não podem iniciar a comunicação nem podem possuir fonte interna de energia.

-Memoria: Ate 65KB de memoria para escrita/leitura.

-Controle autenticado de acesso.

Classe 3 Etiquetas classe 3 possuem sensores acoplados e uma memória interna que suporta gravação de dados provenientes dos sensores como temperatura, pressão e movimento com a bateria interna sem a necessidade da energia provinda da leitora. Mas trata-se de uma etiqueta semi-passiva, possuindo as seguintes características:

-Semi-passiva. Não pode iniciar a comunicação, porem possui sua própria fonte de energia. Permanece passiva ate receber o sinal da leitora.

-Memoria: Até 65KB de memória para escrita/leitura.

-Circuito de sensores integrado.

Classe 4 Etiquetas classe 4 são ativas e possuem um transmissor integrado. São utilizadas para grandes distâncias ou para objetos de grande valor. Devem possuir as seguintes características:

-Ativa. Pode iniciar a comunicação devido a sua bateria interna que alimenta seus circuitos e seu transmissor para gerar o sinal.

-Memória: Deve ser regravável.

-Comunicação: Habilidade de se comunicar com outras etiquetas classe 4.

-Rede. Pode estabelecer uma rede Ad Hoc.

Classe 5 Possuem as mesmas características das etiquetas classe 4 com uma capacidade adicional de comunicação com todas as classes de etiquetas. Por esse motivo, ela também é chamada de etiqueta leitora.

2.4.2 Leitoras

2.4.2.1 Conceito e Características

As leitoras são responsáveis por detectar, ler e escrever nas etiquetas. Elas enviam e recebem sinais através de suas antenas, podendo processar as informações da etiqueta e passar para um computador ou uma rede de computadores. As leitoras tem também a capacidade de enviar ondas de rádio para alimentar as etiquetas, no caso delas serem passivas. [5]

Algumas leitoras mais complexas possuem ainda a capacidade de realizar tratamento anti-colisão, que trata-se de uma medida para garantir a leitura e escrita simultânea de diversas etiquetas. Existem três tipos de técnicas de anti-colisão: espacial, frequência e no domínio do tempo. Todas as três funcionam de forma a estabelecer uma hierarquia ou ordem para que cada etiqueta possa ser comunicar uma de cada vez.

Outra característica de leitoras mais avançadas é a de autenticação. Isso ocorre para leitoras que recebem informações da etiqueta que devem ser autorizadas para evitar fraudes. Geralmente, a etiqueta de RFID fornece um código de chave para a leitora, que passa por um algoritmo para determinar se a chave permite o acesso ao sistema. Existem dois tipos de autenticação: mutual symmetrical e derived keys.

Para proteger a integridade dos dados, algumas leitoras possuem a função de criptografia dos dados. Elas são responsáveis pela criptografia e descriptografia, Isso impede usuários indesejados tenham acessos a informações mesmo acessando o fluxo de dados.

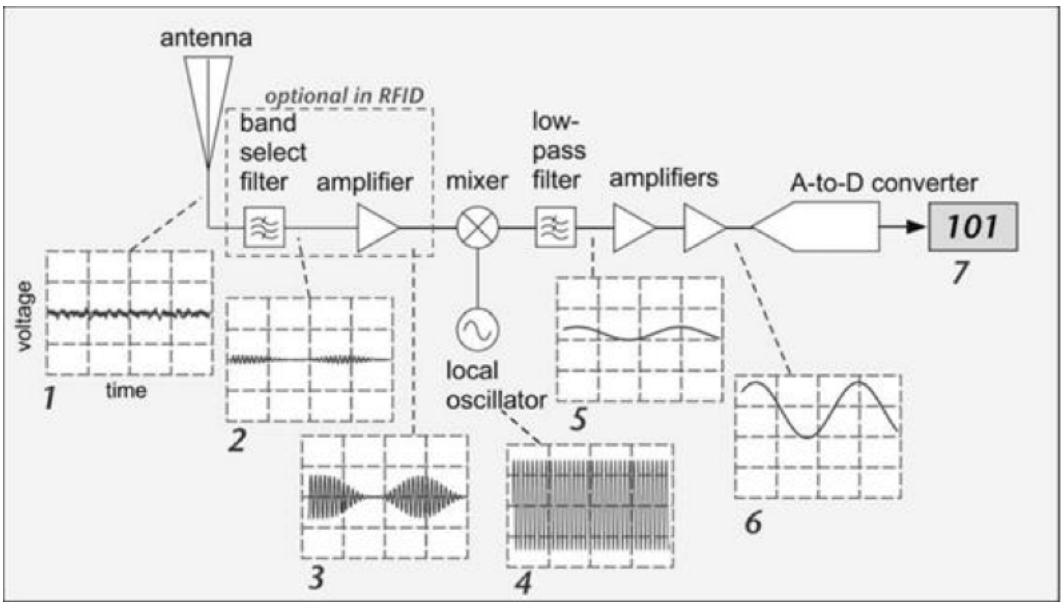


Figura 2.16: Esquemático de uma Leitora RFID

2.4.2.2 Funcionamento

As leitoras se recebem e passam informações com as etiquetas através de uma comunicação com suas antenas. O módulo responsável por essa comunicação se chama transceptor. Ele possui dois módulos: um transmissor e outro receptor. O primeiro transmite energia AC e requisições para as etiquetas na zona de leitura. O receptor recebe sinais analógicos das etiquetas, por meio da antena, e envia esses sinais ao microprocessador para serem digitalizados.

O sinal recebido pode ter inúmeras ondas de rádio indesejadas (1). Por isso, filtros são colocados para selecionar uma banda de operação (por exemplo, 902,5 MHz -907 MHz no Brasil), e eliminar a maioria dos sinais fora desse intervalo de interesse (2). O sinal então é amplificado (3) e misturado com um sinal de frequência e amplitude constantes gerados pelo oscilador (4). Remova-se sinais com frequência superior a frequência de corte com filtro passa-baixa, resultando em um sinal cuja amplitude reflete a intensidade do sinal médio do sinal de alta frequência: o envelope do sinal (5). O sinal passa por um amplificador (6) e, em seguida, por um conversor analógico-digital. [4]

2.4.3 Antenas

Antenas são a parte de um sistema de transmissão ou recepção responsável por radiar ou absorver ondas eletromagnéticas. Ela transforma as ondas guiadas pela linha de transmissão em ondas irradiadas no espaço livre, e vice-e-versa. Em sistemas RFID, existem dois esquemas de instalação de antenas: o monostatic e o bistatic. O monostatic usa apenas uma antena para a transmitir e receber. Já o bistatic possui um antena para transmissão e outra para recepção. Há diversos parâmetros que diferenciam uma antena de outra, como: frequência de ressonância, impedância, ganho, diagrama de irradiação, polarização, eficiência e largura de banda. [10]

2.4.3.1 Diagrama de irradiação

É o mapeamento da distribuição de energia irradiada. As medidas são feitas girando a antena de forma a descrever um círculo em um espaço aberto. Elementos estranhos podem causar interferência, pois causam reflexões, refrações e absorções das ondas. Em intervalos regulares, são realizadas medidas por outra antena, localizada a uma distância não inferior a dez vezes o comprimento de onda. Devido a lei da reciprocidade, o diagrama de transmissão é o mesmo da recepção.

Assim, através dos resultados pode-se analisar as especificações da antena em relação ao campo, à potencia em cada direção, e se sua polarização é vertical ou horizontal.

2.4.3.2 Impedância

A impedância de entrada de uma antena depende da frequência e não pode ser descrita por uma expressão analítica simples. Porém, é possível é possível reapresentá-la como uma resistência em séria com uma reatância para uma frequência constante.

Capítulo 3

Desenvolvimento

3.1 Proposta Inicial do Projeto

A proposta inicial era desenvolver aplicativos em dois celulares (um conectado ao supervisório e outro pessoal do usuário). O celular conectado com o supervisório recebe as informações do ID e da localização do computador, e o cronograma e as opções de alarme do outro celular. Então, envia a localização, e os devidos alertas pro novo celular. No celular pessoal, há uma lista de objetos, onde o usuário define o nome, os horários e dias em que ele sai com o objeto, as opções de alarme(sem alarme, alarme quando sair de casa sem o objeto nos dias definidos, alarme quando o objeto sai sem sua presença), e acessa uma imagem que mostra o cômodo onde encontra-se o objeto. Assim, todo o controle, a localização e a segurança seriam realizadas pelo celular do usuário.

A comunicação dos celulares deveria ser realizada via internet para que o usuário pudesse receber as notificações há longas distâncias. Os celulares utilizados para o desenvolvimento dos aplicativos possuem o sistema operacional Android. Ele oferece como sistema de comunicação pela internet, o SIP (Session Initiation Protocol). Porém, alguns celulares não oferecem suporte SIP, o que era o caso dos celulares onde se estava desenvolvendo o software.

3.2 Projeto do aplicativo

Decidiu-se então buscar uma solução alternativa que houvesse resultados similares. Desse modo, foi desenvolvido apenas um único aplicativo no celular conectado ao supervisório. Nele o usuário define o nome, o fluxo de saída e as opções de alarme dos objetos. Ele recebe informações do ID do objeto quando esse passa pela antena de saída da residência e, de acordo com o fluxo de saída e as opções de alarme definidas, o celular envia uma notificação ao celular pessoal do usuário via SMS.

A localização ficou a cargo do programa desenvolvido no computador. Tal programa se conecta com a leitora e com o celular. Ele mostra a lista de etiquetas, relacionadas com seu respectivo objeto, e quando selecionada mostra sua localização e o histórico de movimentação no ambiente.

3.3 Desenvolvimento do aplicativo

O aplicativo foi desenvolvido para o sistema operacional Android versão 2.3.3 ou superior. Para isso, foi exigido conhecimentos de programação Java e XML. A estratégia utilizada foi desenvolver cada atividade seguindo os seguintes passos:

1. definir interface gráfica (layout e seus componentes);
2. definir ações e permissões no manifesto;
3. relacionar variáveis do aplicativo java aos componentes;
4. desenvolver função de cada componente.

Em todas as atividades, os passos 2 e 3 são semelhantes. O passo 2 é realizado da seguinte maneira. Primeiramente, é definido o nome da atividade, que é o mesmo nome da classe presente no pacote do aplicativo. Em seguida, é definida uma ação, pela qual a atividade será iniciada. O padrão utilizado foi usar o nome do pacote mais o nome da classe em caixa alta. Finalmente, é colocada a categoria da atividade. Tirando MainActivity, que se trata da atividade inicial, todas as outras categorias são definidas como mostra a figura 3.1.

```
</activity>
        <activity
    android:name="erik.wh2.DefAlarme"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="erik.wh2.DEFALARME" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

Figura 3.1: Definição de uma Atividade no Manitesto

As permissões utilizadas por esse aplicativo são: ler calendário, acesso a internet e enviar SMS.

O passo 3 é realizado sempre na fase OnCreate() da atividade. Primeiramente, a atividade é relacionada com o seu respectivo layout através do método setContentView (figura 3.2). E em seguida cada componente do programa é relacionado do método findViewById com os componentes da interface gráfica, encontrados através de seus repectivos ids (figura 3.3).

Para cada atividade, será mostrado o desenvolvimento dos passos 1 e 4.

3.3.1 Classe Objetos

Trata-se de uma classe criada para formar um banco de dados SQL dos objetos com etiquetas. O banco de dados criado possui5 colunas: RowId(número da linha do banco de dados), EPC da etiqueta, nome do objeto, definição do fluxo de saída e opções de alarme.

```
public class DefCrono extends Activity implements OnClickListener,OnCheckedChangeListener{

    TextView crono;
    RadioGroup selection;
    RadioButton semana,especifico;
    CheckBox dom,seg,ter,qua,qui,sex,sab;
    DatePicker esp;
    Button salvar,cancelar;
    String nome,rowId;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.def_crono);
        inicializaVariaveis();
```

Figura 3.2: Definição de Variáveis e Método onCreate

```
private void inicializaVariaveis() {
    // TODO Auto-generated method stub
    crono=(TextView) findViewById(R.id.tvDefCrono);
    selection=(RadioGroup) findViewById(R.id.rgDefCrono);
    semana=(RadioButton) findViewById(R.id.rbSemana);
    especifico=(RadioButton) findViewById(R.id.rbEspecifico);
    dom=(CheckBox) findViewById(R.id.cbDom);
    seg=(CheckBox) findViewById(R.id.cbSeg);
    ter=(CheckBox) findViewById(R.id.cbTer);
    qua=(CheckBox) findViewById(R.id.cbQua);
    qui=(CheckBox) findViewById(R.id.cbQui);
    sex=(CheckBox) findViewById(R.id.cbSex);
    sab=(CheckBox) findViewById(R.id.cbSab);
    esp=(DatePicker) findViewById(R.id.dpEspecifico);
    salvar=(Button) findViewById(R.id.btSalvar);
    cancelar=(Button) findViewById(R.id.btCancelarCrono);
    salvar.setOnClickListener(this);
    cancelar.setOnClickListener(this);
    selection.setOnCheckedChangeListener(this);

}
```

Figura 3.3: Inicialização de Variáveis

3.3.2 MainActivity

Trata-se da atividade inicial do aplicativo.

3.3.2.1 Interface Gráfica

Essa atividade possui um Button nomeado USB e um TestView no topo. E abaixo disso um ListView, com a lista de objetos que o possuem uma etiqueta. O layout foi nomeado activity_main.xml (figura 3.4).

3.3.2.2 Classe Java

Na fase da atividade OnCreate(), é adicionado uma função ao Button USB: quando ele é clicado, ele inicia uma conexão socket com o computador via USB. Foi utilizada uma porta aleatória de número tcp=38300. O aparelho celular roda como servidor e o computador como cliente.

O celular recebe do computador o EPC de cada etiqueta que passa pela antena da saída do laboratório. O aplicativo então inicia uma nova Thread para cada etiqueta. Quando a etiqueta de objetos passam, o seguinte procedimento é realizado:

Figura 3.4: Atividade MainActivity

1. É verificado se o EPC do objeto já existe no banco de dados. Em caso negativo, ele é adicionado com o nome igual ao EPC.
2. São acessadas as opções de alarme do banco de dados;
3. Caso exista alarme em caso de saída do objeto sem a presença do usuário, o programa verifica a presença da etiqueta do usuário através da variável booleana usuário.
4. Em caso positivo, é adicionado o rowId do objeto a uma string objetosComUsuario;
5. Em caso negativo, é esperado 5 segundos e verifica-se a presença do usuário novamente.
6. Em caso positivo, é adicionado o rowId do objeto a uma string objetosComUsuario, separados pelos caracteres":::";
7. Em caso negativo, significa que o objeto está deixando o laboratório sem a presença do usuário, então uma mensagem é enviada para o celular pessoal notificando desse acontecimento.

Quando a etiqueta do usuário é informada, o seguinte procedimento é adotado:

1. Modificar a variável usuário para true para informar sua presença;
2. Esperar 5 segundos;
3. Verificar todos os objetos que devem estar com o usuário na data presente. Para isso, o banco de dados é acessado. São selecionados todos os objetos que possuem opção de alarme em caso de saída do usuário sem a sua presença. Então, os seus respectivos fluxos de saída são verificados se incluem a data atual. Em caso, positivo, seus rowIds são adicionados a listaObjetosDoDia, separados pelos caracteres ":::".

4. As strings com a lista de objetos necessários no dia e a lista de objetos com o usuário são separadas através do método Split("...");

5. Cada objeto necessário no dia é verificado se é encontrado na lista de objetos com o usuário;

6. Em caso positivo, a string com o rowId do objeto necessário no dia é definida como null;

7. No final, é verificado se existe alguma String rowId de objeto necessário no dia diferente de null. Em caso positivo, significa que o objeto necessário não encontra-se em posse do usuário e uma notificação é enviada para o celular pessoal do mesmo.

Na fase OnResume(), é definida a lista de objetos. Foi definida nesta fase da atividade, pois os nomes podem ser alterados por atividades posteriores e no retorno o nome deve ter sido alterado. O programa acessa o banco de dados Objetos, pegando todos os nomes e os respectivos rowIds separados por uma vírgula. Então é utilizado o método slipt(","), e criado dois array de Strings com nome e rowId de cada objeto. Através desse array, é criado o ListView.

Então, é adicionado um ClickListener na lista de objeto. Para que quando um objeto seja clicado possam ser alteradas sua configurações. Então, ao clicar em um objeto, é iniciada atividade DefObjeto e passada pra ela as informações do nome e do rowId do mesmo.

3.3.3 DefObjeto

É uma atividade que define configurações de cada objeto como: nome; fluxo de saída; opções de alarme.

Figura 3.5: Atividade DefObjeto

3.3.3.1 Interface Gráfica

Essa atividade possui um TextView com o nome do objeto. Quatro Button: um para alterar o nome, outro para configurar fluxo de saída, outro para configurar opções de alarme, outro para apagar o objeto do banco de dados. E mais um TestView para mostrar o fluxo de saída atual. O nome do arquivo é def_objeto.xml (figura 3.5).

3.3.3.2 Classe Java

Em OnCreate(), a atividade recebe o nome e o rowId do objeto. Cada Button inicia uma atividade diferente. O que altera nome inicia AlteraNome. O que configura fluxo de saída inicia DefCrono e passa as informações do nome e rowId do objeto. O que configura o alarme inicia DefAlarme e também passa o nome e o row Id. O botão "Apagar Objeto" inicia um método da classe Objetos que recebe o rowId do objeto e apagar essa linha do banco de dados. Em seguida, encerra a atividade.

Em OnResume(), o nome é colocado no TextView. Ele não é colocado na fase OnCreate(), pois existe a possiblidade dele ser alterado pela atividade AlteraNome. No caso de haver alteração, o nome é alterado no banco de dados também.

Em seguida, através do rowId informado, o banco de dados é acessado para requisitar o fluxo de saída do objeto. Como verificaremos na atividade DefCrono, há duas opções de fluxo de saída: dia(s) da semana, ou um ddata específica. No caso da primeira é salvo uma String =”0:::” e no caso da segunda é salvo uma String =”1:::”. No caso da primeira, se o dia da semana é escolhido então o caracter é o 1, se não, o caracter é 0. Começando pela primeira posição sendo o domingo. Assim, uma String =”0:::0101010”, significa que foi escolhido a opção dia(s) da semana, e foram escolhidos os dias segunda-feira, quarta-feira e sexta-feira. No caso da segunda, em seguida vem a data específica no formato (dd/mm/aaaa).

Então, primeiro é a realizada um Split(“:::“) da String obtida do banco de dados. No caso da opção ser 0, são colocados todos os dias da semana em que o caracter é igual a 1. No caso da opção ser 1, é postado a data específica.

3.3.4 AlteraNome

Atividade que altera o nome de um objeto.

3.3.4.1 Interface Gráfica

AlteraNome possui EditText para colocar o nome para o qual deseja alterar e dois Buttons: uma para confirmar e outro para cancelar. O nome do arquivo é altera_nome.xml (figura 3.6). A diferença dessa atividade é que ela foi postada no manisfeto com o tema Dialog. Isso faz com que ela não preencha a tela inteira.

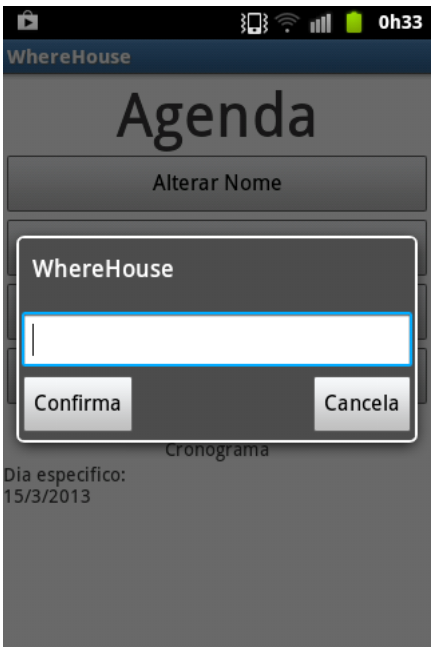


Figura 3.6: Atividade AlteraNome

3.3.4.2 Classe Java

Em OnCreate(), define os ClickListeners de cada Button. Ao clicar em cancelar, a atividade é encerrada. Ao clicar em confirmar, é retirado o texto do EditText e passado para a atividade DefObjeto e então a atividade é encerrada.

3.3.5 DefCrono

É a atividade que define o fluxo de saída do objeto selecionado.

3.3.5.1 Interface Gráfica

TextView para o nome do objeto. Um RadioGroup com dois RadioButton. RadioButtons são componentes onde você seleciona apenas um exclusivamente. Um RadioButton para a opção dia da semana e outro para a opção dia específico. Possui também 7 CheckBox, um para cada dia da semana e um DatePicker para o dia específico. Para finalizar, dois Button: um para confirmar e outro para cancelar. O nome do arquivo é def_crono.xml (figura 3.7).

3.3.5.2 Classe Java

Em OnCreate(), recebe o nome e o rowId do objeto. Então, passa o nome para o TextView. Em seguida, desabilita todos os CheckBox e o DataPicker.

Essa classe implementa OnCheckedChangeListener, que implementa os procedimentos em caso de cada RadioButton ser selecionado. Em caso do dia da semana ser selecionado, todos os Check-

Figura 3.7: Atividade DefCrono

Box são habilitados e o DataPicker é desabilitado. No caso do dia específico ser selecionado, os CheckBox são desabilitados e o DataPicker é habilitado.

A classe também implementa OnClickListener para os Buttons. Se o cancelar for clicado a atividade simplesmente encerra. Se o botão confirmar for clicado, uma string é salva no banco de dados no linha do rowId especificado, na coluna do fluxo de saída. Essa string é salva de acordo com o que foi explicado na atividade DefObjeto. Em caso do RadioButton do dia da semana estar selecionado é salvo a String =”0:::”, seguida de uma sequencia de 0 e 1, de acordo com os CheckBox selecionados. No caso do RadioButton do dia específico, a String salva é igual a =”1:::” seguida da data do DataPicker. E,então, a atividade é encerrada.

3.3.6 DefAlarme

É a atividade que define as opções de alarme.

3.3.6.1 Interface Gráfica

TextView para o nome do objeto; dois CheckBox: um para o alarme em caso de ausência do objeto na saída do usuário e outro para notificar a saída do objeto sem a presença do usuário; dois Button: confirmar, cancelar. O nome do arquivo é def_alarme.xml (figura 3.8).

3.3.6.2 Classe Java

Em OnCreate(), recebe o nome e o rowId do objeto. Passa o nome para o TextView. Acessa o banco de dados na linha do rowId e requisita as opções de alarme. Ele salva uma string de dois

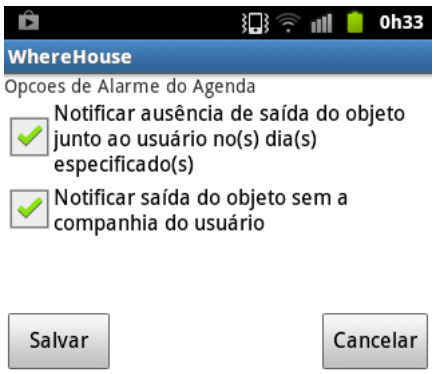


Figura 3.8: Atividade DefAlarme

caracteres. Se o primeiro carácter for 0 então a opção notificar em caso de ausência do objeto na saída do usuário está desmarcada. Se for 1, está marcada. Se o segundo caracter for 0 então a opção notificar a saída do objeto sem a presença do usuário está desmarcada. Se for 1, está marcada.

A classe implenta OnClickListener. No caso do Button cancelar ser clicado, a atividade é encerrada. No caso do Button confirmar for clicado, o aplicativo salva o fluxo de saída na linha rowId, através de uma String de dois caracteres como foi explicado acima. Então, a atividade é encerrada.

3.4 Programa Supervisório

O programa deve realizar as seguintes tarefas:

• Conectar-se com a leitora - Receber da leitora EPC de cada etiqueta lida e a antena responsável pela leitura;

• Conectar-se com o celular - Enviar o EPC de cada etiqueta lida para o celular e receber o nome do objeto da etiqueta;

• Mostrar uma lista de todos as etiquetas lidas

• Quando selecionada uma etiqueta, mostrar uma imagem com sua localização e o histórico de sua movimentação.

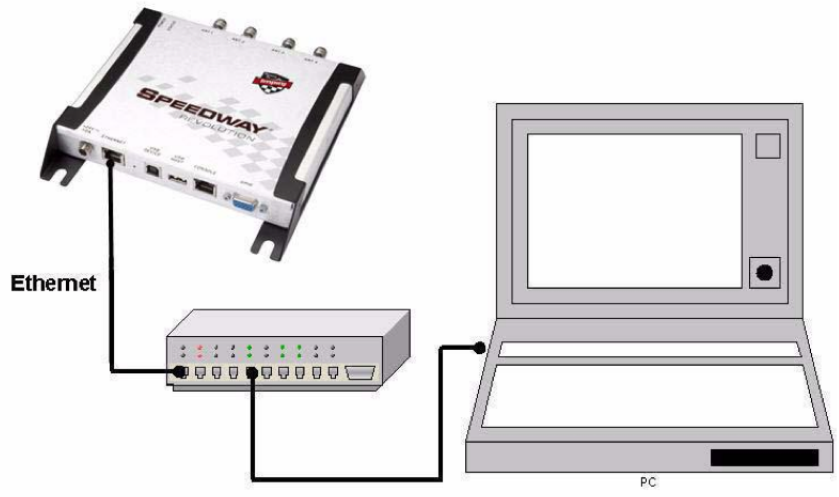


Figura 3.9: Conexão com a leitora via TCP/IP

3.4.1 Conexão com a leitora

Para se conectar com a leitora foi utilizado o código JAVA de amostra fornecido pela impinj (http://learn.impinj.com/articles/en_US/RFID/Creating-RFID-Applications-with-Java/). O programa abre a conexão com a leitora através do protocolo tcp/ip (figura 3.9), ativa todas as antenas, espera 50 segundos e fecha a conexão e desativa as antenas. De forma assíncrona, se alguma etiqueta é lida, o programa imprimi na tela a EPC da etiqueta e o tempo de voo da sua leitura.

Então, utilizamos esse programa como uma classe nomeada HelloJavaLtk. Foi retirada a espera de 50 segundos para que a conexão com a leitora se encerre apenas com o fechamento do programa. Criou-se uma variável global para assumir o número da EPC da etiqueta quando uma é lida (String epcEnviado) e outra para assumir o número da antena que realiza a leitura(String antenas).

Na classe principal, depois de realizada a conexão, o programa verifica se a variável epcEnviado é diferente de null, o que significa que uma etiqueta foi lida. Quando isso ocorre, ele verifica a existência da etiqueta nos arquivos.

-Em caso negativo, ele abre o arquivo arquivo.tmp na pasta res e grava a nova etiqueta. Tal arquivo agrupa todas as etiquetas separadas pelo caractere "!". E as etiquetas são definidas pelo epc, seguido pelos caracteres ":::", e depois pelo nome do respectivo objeto. O programa então pergunta abre uma janela (figura x) perguntando qual a atual posição do objeto (A1, A2, A3 ou A4). Então, cria-se um novo arquivo com o EPC da etiqueta nas pasta res/histórico com a data e a hora da leitura e a posição dada, separados com os caracteres ":::".

-Em caso afirmativo, abre-se o arquivo da pasta res/histórico com o epc da etiqueta lida e adiciona-se ao histórico a nova posição e a data e hora da nova leitura. Pela figura XX, verificamos que a antena 1 está entre os setores A1 e A3, a antena 2 está entre A2 e A4, a antena 3 está entre A3 e A4 e a antena 4 entre A3 e o exterior. A nova posição depende da posição anterior e de qual antena realiza a leitura. A antena é passada pela String antenas da classe HelloJavaLtk. A posição anterior é retirada da última posição no arquivo do histórico. Então, a análise é feita da seguinte

forma de acordo com a antena que realizou a leitura:

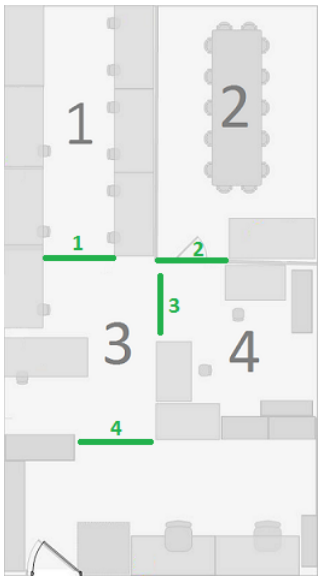


Figura 3.10: Conexão com a leitora via TCP/IP

- Antena 1: Se a posição anterior era A3, então a posição atual é A1, caso contrário a posição atual é A3.
- Antena 2: Se a posição anterior era A4, então a posição atual é A2, caso contrário a posição atual é A4.
- Antena 3: Se a posição anterior era A3, então a posição atual é A4, caso contrário a posição atual é A3.
- Antena 4: Se a posição anterior era A3, então a posição atual é A0 (Fora do ambiente), caso contrário a posição atual é A3.

No caso do objeto se mover da posição A3 para A0 significa estar saindo do ambiente, então ele envia o EPC para o celular pela porta USB.

3.4.2 Conexão com o celular

Para se conectar com o celular foi utilizada uma conexão socket definindo uma porta para enviar e receber dados entre o celular e o computador. No caso, a porta escolhida de forma aleatória foi a de número 38000. A classe responsável por essa função foi nomeada EchoClient. Ela possui dois métodos: conexao() e enviaUSB(String EPC).

O primeiro realiza a conexão socket pela porta 38300, inicializando as variáveis responsáveis pelos dados de entrada e de saída.

O segundo recebe uma string com o número EPC de uma etiqueta e então envia esse dado para o celular e recebe do mesmo o nome dado pelo usuário no aplicativo. Esse nome é gravado no arquivo.tmp da pasta res.

3.4.3 Interface Gráfica

O layout usado para a interface gráfica do programa utilizado foi o BorderLayout. Tal layout define regiões como norte, sul, centro, leste e oeste para organizar os componentes como mostra a figura 3.11.

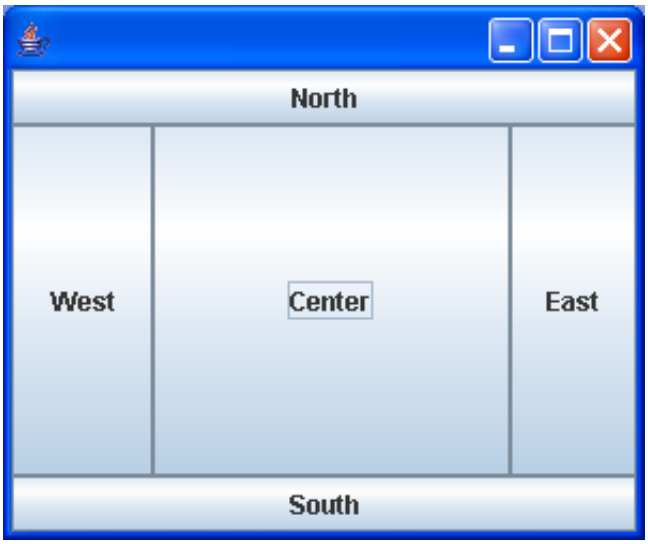


Figura 3.11: Modelo de um Border Layout

No norte do BorderLayout foi inserido um barra de menu. Nela estão presentes dois componentes: wherehouse e ajuda. O wherehouse possui 4 itens: Conectar USB, Conectar Leitora, Posicionar Objeto e Sair. O ajuda possui apenas um item: Sobre. A seguir, são descritas as funções de cada item:

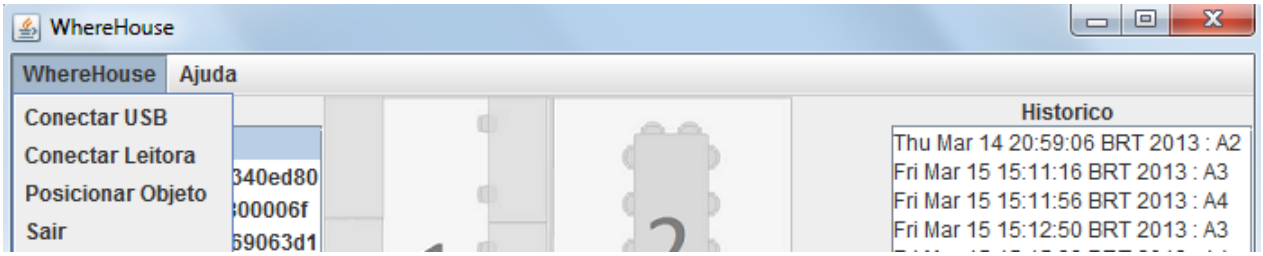


Figura 3.12: Barra de Menu I

-Conectar USB

Inicia uma Thread chamada USBConectado. Ela instancia a classe EchoClient inicia o método conexão dessa classe.

-Conectar Leitora

Inicia uma Thread chamada Leitora, que instancia a classe HelloJavaLtk e conecta-se com a leitora.

-Posicionar Objeto

Serve para posicionar o objeto manualmente, caso sua posição esteja incorreta. Abre uma janela para que o usuário entre com a posição atual e então salva no arquivo histórico do objeto, a data ,a hora atual e a sua nova posição.

-Sair

Fecha o programa e encerra todas as conexões.

-Sobre

Abre uma janela sobre os responsáveis pelo desenvolvimento do programa (figura 3.13).

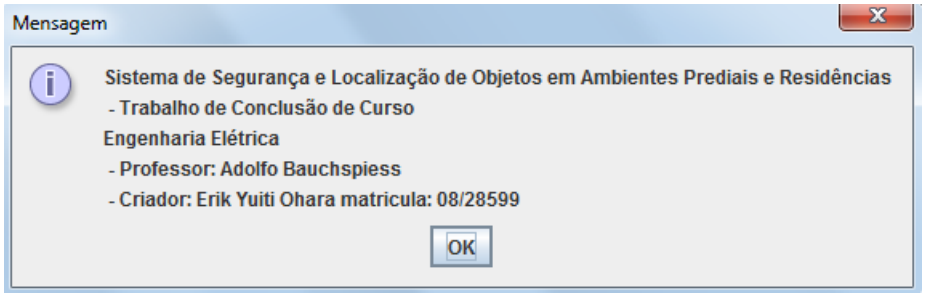


Figura 3.13: Janela Sobre do Programa

Na borda oeste do programa, é inserido uma lista das etiquetas lidas pela leitora. Abre-se o arquivo.tmp da pasta res. Utiliza-se o método slipt pelo caractere "!", obtendo dois vetores, um contendo os nomes dos respetivos objetos e outro os EPC das etiquetas. Então, é criado uma lista pelo nome dos objetos (em caso de não serem nomeados, o nome dado é o EPC da etiqueta). Quando selecionado algum objeto, esse abre uma imagem no centro e o seu histórico no leste.

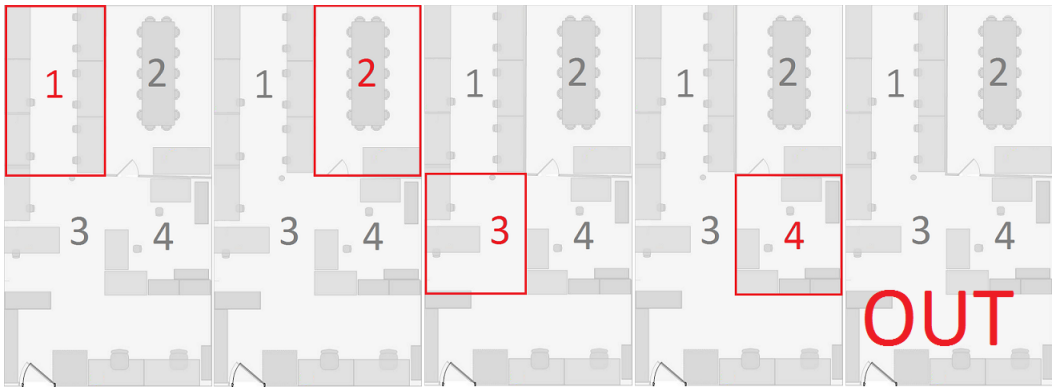


Figura 3.14: Imagens do posicionamento dos objetos

A imagem é aberta através da classe LoadImage que carrega uma imagem presente na pasta res/img. Dependendo da posição atual lida no arquivo do histórico, uma imagem diferente é carregada. As imagens são mostradas pela figura 3.14.

O histórico no leste é mostrado em uma área de texto, mostrando a data e hora da leitura e a posição para a qual o objeto se deslocou.

No sul, é inserido um texto apenas para mostrar o status de conexão do programa com o celular e com a leitora.

Assim, o a interface gráfica ficou da seguinte forma:

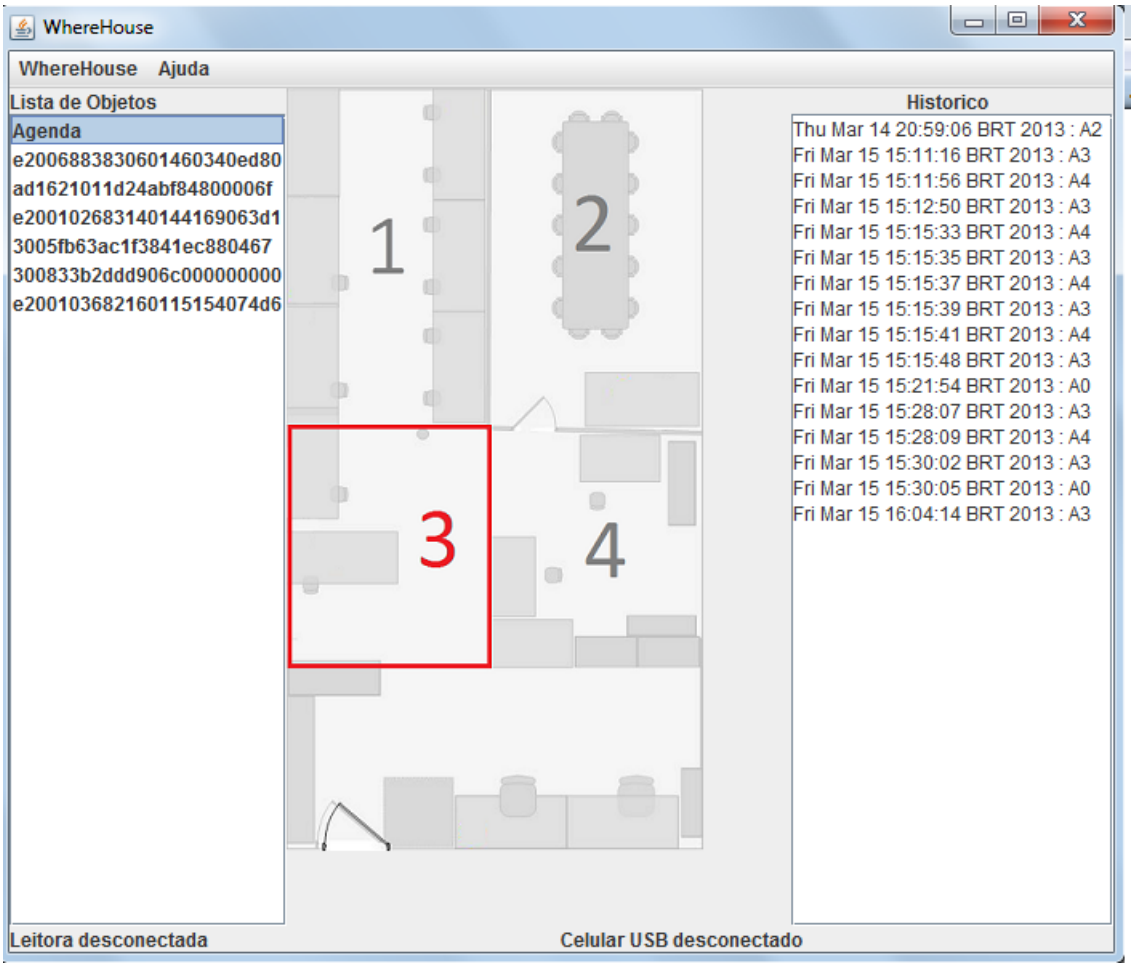


Figura 3.15: Imagens do posicionamento dos objetos

Capítulo 4

Resultados experimentais e Análise dos dados

Nesse capítulo, serão mostrados os resultados de simulações realizadas para verificar a funcionalidade do sistema, e serão analisados os resultados obtidos. Também irá ser verificado a possiblidade do projeto em questão ser aplicado em uma residência. Serão considerados valor dos objetos, preço de seguros residenciais e custo dos equipamentos.

4.1 Simulação 1

A primeira simulação foi realizada dia 14/04/2013 no Laboratório de Automação e Robótica da Universidade de Brasília. As antenas foram colocadas no chão nos mesmos locais indicados pela figura 3.10. A etiqueta do usuário foi colocada dentro de sua carteira. Foram colocadas etiquetas também nos seguintes objetos: documento do carro, carregador de bateria, pasta e chaveiro.

4.1.1 Resultados

A primeira leitura foi da movimentação de apenas o usuário realizada às 16h08. A seguinte tabela 4.1 mostra a movimentação realizada pelo usuário e o resultado mostrado pelo programa:

Tabela 4.1: Movimentação do Usuário.

Usuário	
Esperado	resultado
A1	A1
A3	A3
A4	A4
A2	A2
A4	A4
A3	A3
A1	

O único erro ocorreu pois não houve a leitura da última movimentação. Vídeo mostra que a carteira não passou por cima da antena pois essa estava mal posicionada.

Em seguida, o usuário realizou o mesmo percurso, mas carregando o documento do carro.

Tabela 4.2: Movimentação do Usuário com o Documento do carro.

Usuário		Documento do carro	
Esperado	Resultado	Esperado	Resultado
A1	A1	A1	A1
A3	A3	A3	A3
A4	A4		A1
A2	A2		A3
A4	A3		A0
A3			A3
A1		A4	A4
		A2	A2
		A4	A4
			A2
		A3	A3
			A4
			A3
		A1	A1
			A3

Em relação a etiqueta do usuário, houveram duas falhas de leitura. Na movimentação da volta de A2 para A4, a antena 2 não realizou a leitura. Assim como na movimentação de A4 para A3. Então, quando a antena 1 realizou a leitura e a posição anterior era A2, a posição atual passou ser A3 e não A1.

Já a etiqueta do documento do carro, ocorreram múltiplas leituras na transação entre os ambientes. Parecendo para o programa que o objeto estava se deslocando entre os cômodos rapidamente. Outra coisa que ocorreu foi que ao passar por A3 na ida, a antena 4 realizou a leitura da etiqueta, fazendo com que o programa concluísse que o objeto deixou o local.

No cenário seguinte, o usuário realizou o mesmo percurso, carregando o documento do carro e o carregador de bateria.

Tabela 4.3: Movimentação do Usuário com o documento do carro e carregador.

Usuário		Documento do carro		Carregador	
Esperado	Resultado	Esperado	Resultado	Esperado	Resultado
A1	A1	A1	A1	A1	A1
A3	A3	A3	A3	A3	
A4	A4	A4		A4	
	A3	A2		A2	A4
	A4	A4		A4	A2
	A3	A3		A3	
A2		A1		A1	
A4					
A3					
A1					

Com o usuário, ocorreram múltiplas leituras na passagem de A3 para A4, e as seguinte leituras não ocorreram.

O documento do carro só foi lido na transição de A1 para A3.

O carregador só foi lido na transação de A4 para A2, onde como a posição anterior era A1, então passou para A4. E na transação de A4 para A2, o programa entendeu que o movimento contrário estava ocorrendo.

Agora, o usuário percorre o mesmo caminho, com a pasta além dos objetos anteriores.

Tabela 4.4: Movimentação do Usuário com o documento do carro, carregador e pasta.

Usuário		Documento do carro		Carregador		Pasta	
Esperado	Resultado	Esperado	Resultado	Esperado	Resultado	Esperado	Resultado
A1	A1	A1	A1	A1	A1	A1	A1
A3	A3	A3	A3	A3		A3	A3
A4	A4	A4		A4		A4	A4
A2	A2	A2		A2		A2	
A4		A4		A4		A4	
A3	A3	A3		A3		A3	
A1	A4	A1		A1		A1	
	A3						

A etiqueta do usuário não foi lida pela antena 2 na volta de A2 para A4. Então ocorreram múltiplas leituras pela antena 3 na transação de A4 para A3. E a antena 1 não leu a etiqueta quando essa passava de A3 para A1.

O documento do carro só foi lido pela antena 1 da primeira mudança de ambiente.

A pasta foi lida pelas duas primeiras antenas do percurso. Em seguida, não houve mas nenhuma registro de sua movimentação.

Em seguida, o usuário se movimenta com 4 objetos: documento do carro, carregador de bateria, pasta e chaveiro como mostra a tabela 4.5.

A etiqueta do usuário novamente não foi lida pela antena 2 na volta de A2 para A4. Também não ocorreu a leitura da antena 1 na última movimentação.

O documento do carro é lido nas duas primeiras antenas do percurso.

O carregador de bateria só é lido pela primeira antena.

A pasta recebe a leitura da antena 4 quando estava presente no ambiente A3. A antena 3 não realizou a leitura. A antena 3 realiza múltiplas leituras e a antena 1 não detecta a pasta na volta.

O chaveiro realiza múltiplas leituras em quase todas as transações. Exceto na volta, onde as antenas 3 e 1 não realizam a leitura.

4.1.2 Análise

Os erros que ocorreram na simulação foram devido aos seguintes fatos:

1. Ausência de leitura de alguns objetos;
2. Leitura indevida de antenas por onde o objeto não passou;
3. Múltiplas leituras.

Tabela 4.5: Movimentação do Usuário com o documento do carro e carregador, pasta e chaveiro.

Usuário		Documento do carro		Carregador		Pasta		Chaveiro	
Esperado	Resultado	Esperado	Resultado	Esperado	Resultado	Esperado	Resultado	Esperado	Resultado
A1	A1	A1	A1	A1	A1	A1	A1	A1	A1
A3	A3	A3	A3	A3	A3	A3	A3	A3	A3
A4	A4	A4	A4	A4		A4	A0	A4	A1
A2	A2	A2		A2		A2	A4		A3
A4		A4		A4		A4	A2		A1
A3	A3	A3		A3		A3	A3		A3
A1		A1		A1		A1	A4		A1
							A3		A3
									A1
									A3
								A4	A4
									A3
									A4
									A3
									A4
									A3
									A4
								A2	A2
								A4	A4
									A2
									A4
									A2
									A4
									A2
									A4
									A2
									A4
								A3	
								A1	

O posicionamento das antenas no chão, faz com que muitas vezes não ocorra a leitura dos objetos, pois se passam distantes das antenas. Isso pode ser melhorado posicionando as antenas de forma vertical nos locais indicados pela figura 3.10.

As leituras indevidas ocorrem porque a potência de transmissão está muito grande ou a sensibilidade está muito pequena. Essas leituras podem ocorrer também devido a obstáculos no meio do caminho. Dependendo da obstrução, o obstáculo pode aumentar a potência recebida pela etiqueta. Porém, houve momentos da simulação em que as etiquetas não foram lidas. Se diminuirmos a potência de transmissão ou aumentarmos a sensibilidade, podemos realmente evitar as leituras indevidas, mas corre-se o risco de aumentarmos a ausência de leituras, tornando o sistema menos eficaz. Principalmente em relação a segurança, esse é um rico que não pode ser corrido. Assim, deve ser mantido a potência de transmissão e a sensibilidade das antenas.

Para solucionar o problema da múltiplas leituras, o programa será modificado de forma que quando ocorra uma leitura, o programa espere 2 segundos até que habilite a leitura dessa mesma etiqueta.

4.2 Simulação 2

O programa foi alterado como foi dito no capítulo anterior e as antenas foram posicionadas de forma vertical.

Dessa vez, o cenário foi definido de forma diferente:

- Movimentações eram realizadas a cada 5 minutos, de forma que poderia ser calculado o tempo total em que o sistema localizaria o objeto de forma correta.
- A cada meia hora houve uma simulação de tentativa de roubo dos objetos.
- A cada uma hora o usuário saia do ambiente para verificar notificação de posse de certos objetos.
- Nas primeiras 4 horas, os usuários movimentavam os objetos em suas mãos
- Da 4ª até a 5ª hora, os objetos foram movimentados dentro de uma mochila
- Da 5ª até a 6ª hora, as etiquetas foram movimentadas dentro dos bolsos dos usuário
- Da 6ª até a 7ª hora, os objetos eram movimentados em posições diferentes como: seguras no alto pelo usuários, com os usuários agachados, com os os usurários rastejando, etc.

O planejamento da simulação 2 está presente no anexo I.

Após análise do planejamento, conclui-se que a movimentação dos objetos será da seguinte forma nas primeiras 4 horas, como mostram as figuras 4.1,4.2,4.3 e 4.4.

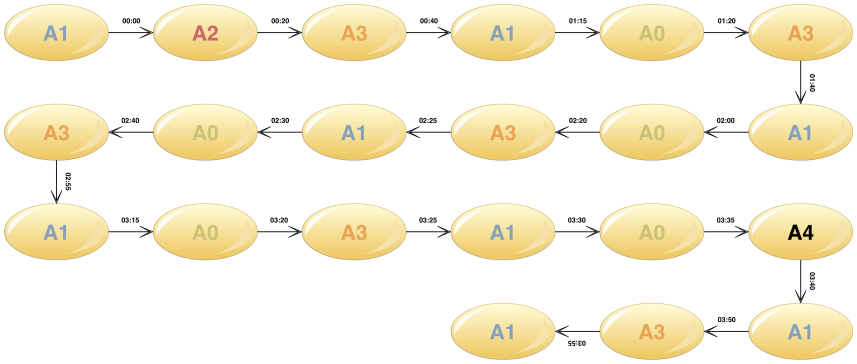


Figura 4.1: Movimentação do objeto Documento do Carro nas primeiras 4h.

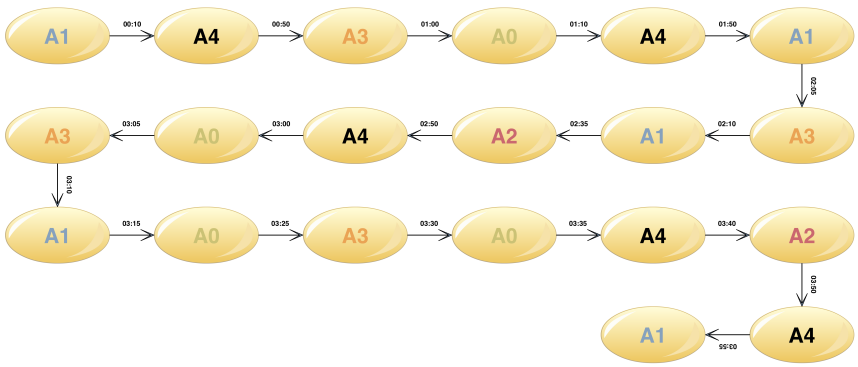


Figura 4.2: Movimentação do objeto Pasta nas primeiras 4h.

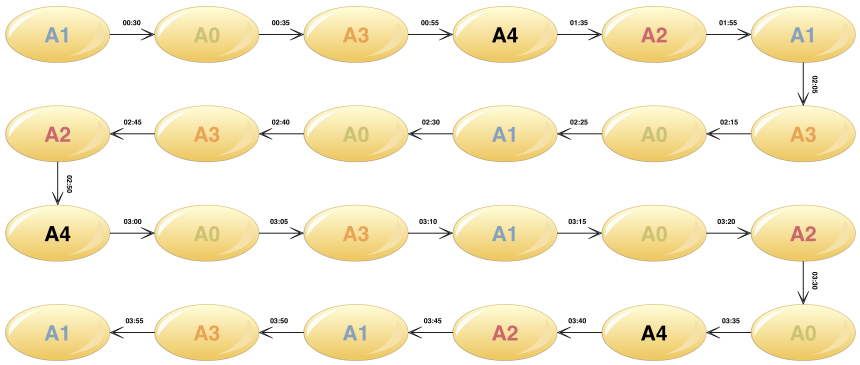


Figura 4.3: Movimentação do objeto chaveiro nas primeiras 4h.

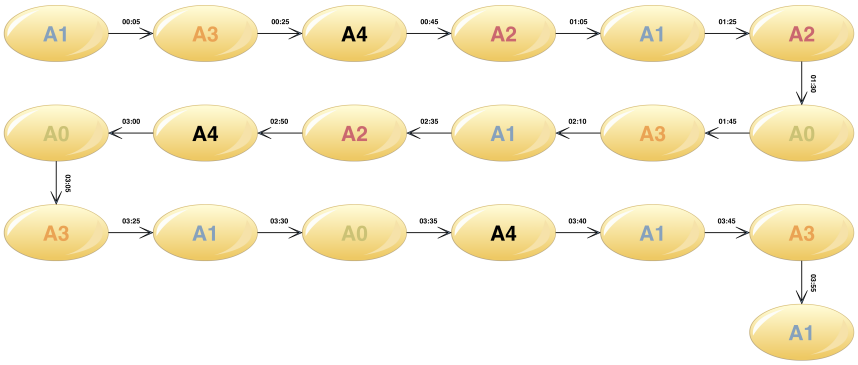


Figura 4.4: Movimentação do objeto carregador de bateria nas primeiras 4h.

A movimentação dos objetos das 4ª até a 5ª hora é mostrada pelas figuras 4.5, 4.6, 4.7 e 4.8. As movimentacões da 5ª até 6ª hora e da 6ª até a 7ª hora são as mesmoas.

Figura 4.5: Movimentação do objeto Documento do Carro da 4ª até a 5ª hora.

Figura 4.6: Movimentação do objeto Pasta da 4ª até a 5ª hora.

Figura 4.7: Movimentação do objeto chaveiro da 4ª até a 5ª hora.

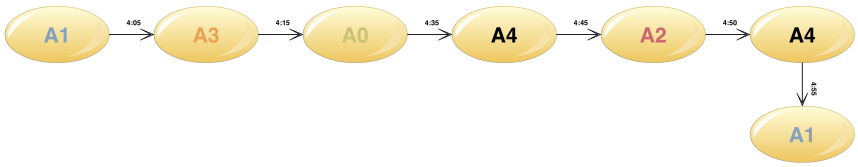


Figura 4.8: Movimentação do objeto carregador da 4ª até a 5ª hora.

4.2.1 Resultados

O histórico dos resultados impressos pelo programa encontram-se no anexo **??**. A partir deles, as tabelas 4.6,4.7, 4.8, 4.9 abaixo foram feitas:

Tabela 4.6: Resultados dos objetos nas primeiras 4h.

Objeto	nº antenas no percurso	nº leituras válidas	nº leituras inválidas	% do tempo que localização estava correta
Documento do carro	30	27	0	85.42 (35 min de posição incorreta)
Pasta	28	27	1	83.33 (40 min de posição incorreta)
Chaveiro	38	35	2	77.08 (55 min de posição incorreta)
Carregador de bateria	30	28	0	72.92 (65 min de posição incorreta)

Tabela 4.7: Resultados dos objetos da 4ª até a 5ª hora.

Objeto	nº antenas no percurso	nº leituras válidas	nº leituras inválidas	% do tempo que localização estava correta
Documento do carro	10	3	0	16.67 (50 min de posição incorreta)
Pasta	10	3	0	33.33 (40 min de posição incorreta)
Chaveiro	10	4	0	41.67 (35 min de posição incorreta)
Carregador de bateria	8	2	0	16.67 (50 min de posição incorreta)

Tabela 4.8: Resultados dos objetos da 5ª até a 6ª.

Objeto	nº antenas no percurso	nº leituras válidas	nº leituras inválidas	% do tempo que localização estava correta
Documento do carro	10	5	0	50.00 (30 min de posição incorreta)
Pasta (só a etiqueta)	10	5	0	8.33 (55 min de posição incorreta)
Chaveiro	10	5	0	16.67 (50 min de posição incorreta)
Carregador de bateria	8	4	0	66.67 (20 min de posição incorreta)

Tabela 4.9: Resultados dos objetos da 6ª até a 7ª.

Objeto	nº antenas no percurso	nº leituras válidas	nº leituras inválidas	% do tempo que localização estava correta
Documento do carro	10	5	0	66.67 (20 min de posição incorreta)
Pasta	10	1	0	33.33 (40 min de posição incorreta)
Chaveiro	10	1	0	41.67 (35 min de posição incorreta)
Carregador de bateria	8	6	0	91.67 (5 min de posição incorreta)

A seguir, encontra-se os resultados das simulações de tentativas de furto mostrados pela tabela 4.10:

Tabela 4.10: Resultados das simulações de furto dos objetos

Período	Tentativas de furtos	nº de Notificações precisas	nº de Notificações falhas
Nas primeiras 4h	13	11	2
Da 4ª a 5ª hora	2	1	0
Da 5ª a 6ª hora	2	0	0
Da 6ª a 7ª hora	2	0	0

Diversas vezes o usuário saiu do ambiente para simular o "esquecimento" de alguns objetos. A tabela 4.11 mostra os resultados dessa simulação.

Tabela 4.11: Resultados das simulações de "esquecimento" dos objetos

Período	nº de notificações necessárias	nº de Notificações enviadas	nº de Notificações falhas
Nas primeiras 4h	9	9	1
Da 4ª a 5ª hora	2	2	2
Da 5ª a 6ª hora	2	2	1
Da 6ª a 7ª hora	2	2	0

4.2.2 Análise

Nas primeiras 4 horas, nota-se que há tanto momentos em que a antena não consegue ler a etiqueta e momentos em que ela le há uma distância muito grande, de forma que o computador interpreta a posição do objeto de forma errada. Desse modo, se a potência de transmissão é aumentada, diminui o número de falhas de leitura, mas aumenta as leituras indevidas.

Primeiramente, analisaremos as caracteríticas das antenas para verificar se alguma delas pode ser a responsável pelo problema. A figura 4.9 mostra o diagrama de radiação das antenas utilizadas.

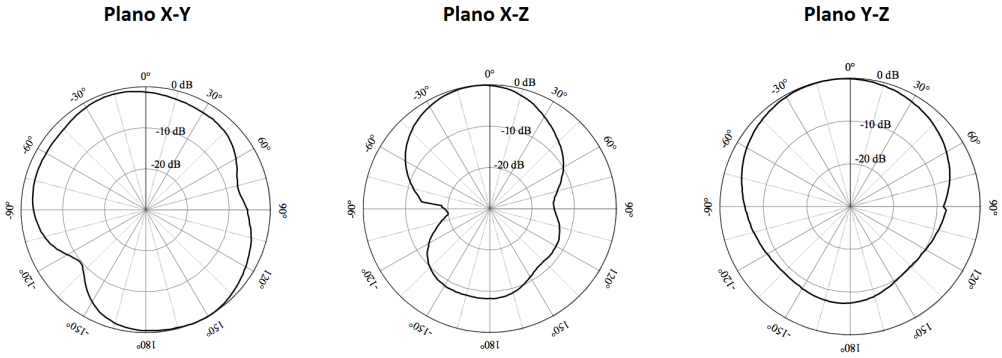


Figura 4.9: Diagrama de Radiação da antenas do sistema

Podemos notar pelo plano X-Z, que trata-se de uma antena bem diretiva. Mas até -15º a potência é quase tão grande quanto a 0º. Isso pode estar causando as leituras indevidas. Todavia uma antena mais diretiva, provavelmente não teria um formato muito adequando para se implantar nesse sistema. O que ocorre é que as antenas 3 e 4 separam ambientes que não possuem divisórias entre si, o que proporciona uma maior probalilidade de leituras indevidas. No caso de residências, geralmente, os cômodos são separados por portas, sendo lugares ideias para o posicionamento de antenas.

Outro problema ocorre quando a etiqueta move-se até a antena e volta para o ambiente anterior, pois o programa coloca sua posição, como se a etiqueta tivesse mudado de posição. Uma solução seria colocar duas antenas em cada divisória, de forma que a leitura consecutiva das duas sigficaria a travessia da etiqueta e a leitura de apenas um significaria a volta dela para o cômodo anterior. Essa

duplicação das antenas também resolveria o problema de programação do erro quando a posição está incorreta. Quando se tem apenas uma antena, a duas possibilidades de destino da etiqueta. O programa resolve esse problema através do conhecimento da posição anterior da etiqueta. Mas, se a posição anterior está incorreta, então a próxima também estará, propagando o erro da localização.

Da 4ª até a 6ª hora, verificou-se que há um grande problema quando se coloca objetos dentro de mochilas ou dentro de bolsos. É necessário que as etiquetas possam ser lidas nesses modos, pois de outra forma a segurança torna-se totalmente ineficaz. Existem algumas etiquetas mais sensíveis que provavelmente devem ser lidas mesmos nessas condições. Infelizmente, não foi possível realizar simulações com essas.

Da 6ª até a 7ª hora, algumas posições se mostraram problemáticas para sua leitura: com o objeto no alto, rastejando e dentro das mãos. A etiqueta dentro das mãos é o mesmo problema da mochila e dos bolsos. O problema do objeto no alto e da pessoa rastejando está no tamanho da antena. É necessário uma antena da altura da divisória dos ambientes para o tornar o sistema eficaz.

4.3 Viabilidade

Para ter uma ideia do valor da segurança de seus objetos foram realizadas duas cotações de seguro residencial: sem e com seguro contra roubo. A diferença é o valor que custa para assegurar seus bens. O valor assegurado foi definido com base em uma em uma pesquisa realizada pela CODEPLAN, onde foi desenvolvida uma tabela de domicílios ocupados segundo a condição de posse de bens. Com isso, foi selecionado os itens com maior possibilidade de serem assaltados. Depois, foi definido a quantidade pelo quanto a maior parte da população possui. Então, buscou-se preços médios em sites de compra. O resultado foi o valor de R$ 5500,00 em posses de bens.

A diferença das cotações com e sem foi de R$ 85.38 (as cotações encontram-se no anexo IV).

A implantação do sistema do LARA foi de R$ (de acordo com a cotação do dólar comercial do dia 18/04/2013). Além de que em residênciais é comum haver mais de quatro ambientes para se monitorar. Também chegamos a conclusão de seria necessário o dobro de antenas para tornar o sistema eficaz. Assim, ainda é inviável tornar este trabalho um produto comercial. Deve ser feito um estudo sobre o desenvolvimento das próprias antenas, além da compra de leitoras mais simples. Desse modo, talvez seja possível diminuir o custo para esta implantação.

Capítulo 5

Conclusões

Nesse trabalho, foi desenvolvido um sistema para localizar, assegurar e controlar objetos em ambientes prediais e residenciais. Foi mostrado o quanto a segurança das posses de cada um é importante. E como a capacidade de localização e controle dos objetos facilitaria o dia-a-dia, que torna-se cada vez mais corrido.

Devido a desenvolvimento de outros projetos sobre RFID, o Laboratório de Automação e Robótica da Universidade de Brasília obteve leitoras e antenas RFID. Assim, era necessário o desenvolvimento de programas para implementar o sistema desejado.

A ideia inicial era que todo a visualização da localização, o controle dos objetos e as notificações de segurança fossem realizadas através de um celular pessoal. Foi escolhido o sistema operacional Android, devido a facilidade de desenvolvimento de aplicativos para essa plataforma. A linguagem de programação utilizada é o Java. Desse modo, a leitora se conectaria ao computador pela rede local e um programa de computador passaria as informações para um celular conectado pela porta USB, que se comunicaria com o celular pessoal do usuário pelo procolo SIP (Session Initiation Protocol).

Ao programar o aplicativo, foi encontrado uma limitação das versões do android que não permitiam o protoco SIP. Então, uma nova alternativa haveria que ser formulada. As notificações então passaram a ser por SMS. E para configurar o fluxo de saída dos objetos e visualizar suas localizações é necessário acessar o computador e o celular conectado ao mesmo.

Com o novo aplicativo desenvolvido, foram realizadas duas simulações do sistema totalizando 12 horas de simulação. Na simulação 1, que durou 5 horas, foram feitos diversos testes para verificar problemas no programa e na implementação do sistema. Na simulação 2, que durou 7 horas, alguns problemas foram corrigidos e foi analisado a quantidade de erros que acontecia da mesma forma. Concluiu-se que o sistema possui diversas limitações, mas foram apresentadas possíveis soluções para serem testadas posteriormente.

Então, foi analisado a viabilidade de um sistema com essas finalidades ser aplicada no mercado consumidor. Através de cotações de seguros contra roubos e análises orçamentárias, foi concluído que o custo para possuir um sistema como o desse trabalho ainda é muito alto para ser tornar

um produto comercial. Porém, é possível que com um estudo mais aprofundado das leitoras e das antenas, o custo diminua, tornando esse trabalho viável. Assim, pessoas poderão assegurar suas posses e também ter a comodidade de localizá-las em suas residências e nunca mais esquecê-las ao deixar sua moradia.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] -http://www.devmedia.com.br/etapas-do-desenvolvimento-e-execucao-de-uma-aplicacao-java/25099.

[2] http://developer.android.com/index.html.

[3] http://www.codeplan.df.gov.br/images/codeplan/pdf/pesquisas

[4] Daniel M. Dobkin. *The RF in RFID.* Newnes, 2008.

[5] Klaus Finkenzeller. *RFID Handbook.* Wiley, 2003.

[6] V. DANIEL HUNT, ALBERT PUGLIA, and MIKE PUGLIA. *RFID a guide to radio frequency identification.* WILEY-INTERSCIENCE, 2007.

[7] Judith M. Myerson. *RFID in the Supply Chain - Guide to Selection and Implemetation.* Auerbach Publications, 2007.

[8] Herb Schildt and Patrick Naughton. *The complete reference Java 2.* OSBORNE, 4nd edition, 1999.

[9] Kathy Sierra and Bert Bates. *Sun Certified Programmer for Java 2 Platform.* OSBORNE - MCGRAW- HIL, 3nd edition, 2003.

[10] Warren L. Stutzman and Gary A. Thiele. *Antenna Theory and Design.* John Wiley e Sons, INC, 2nd edition, 1998.

ANEXOS

I. PLANEJAMENTO DA SIMULAÇÃO 2

Movimentação de usuários com os objetos em suas mãos na altura da cintura. Todos os objetos começaram na ambiente A1. Todos os fluxos de saída dos objetos incluem a data atual.

Tabela I.1: Planejamento da simulação 2 nas primeiras 4 horas.

Horário	Movimentações
00:00	Documento do carro é movimentado para A2
00:05	Carregador de bateria é movimentado para A3
00:10	Pasta é movimentada para A4
00:15	Usuário sai sem nenhum objeto
00:20	Documento do carro é movimentado para A3
00:25	Carregador de bateria é movimentado para A4
00:30	Roubo do chaveiro
00:35	Chaveiro é movimentado para A3
00:40	Documento do carro é movimentado para A1
00:45	Carregador de bateria é movimentado para A2
00:50	Pasta é movimentada para A3
00:55	Chaveiro é movimentado para A4
01:00	Roubo da pasta
01:05	Carregador de bateria é movimentado para A1
01:10	Pasta é movimentada para A4
01:15	Usuário sai apenas com o documento do carro
01:20	Documento do carro é mocimentado para A3
01:25	Carregador de bateria é movimentado para A2
01:30	Roubo do carregador de bateria
01:35	Chaveiro é movimentado para A2
01:40	Documento do carro é movimentado para A1
01:45	Carregador de bateria é movimentado para A3
01:50	Pasta é movimentada para A1
01:55	Chaveiro é movimentado para A1
02:00	Roubo do documento do carro
02:05	Pasta e chaveiro são movimentados para A3
02:10	Carregador de bateria e pasta são movimentados para A1
02:15	Usuário sai com o documento do carro e chaveiro
02:20	Documento do carro é movimentado para A3
02:25	Documento do carro e chaveiro são movimentados para A1
02:30	Roubo do documento do carro e do chaveiro
02:35	Carregador de bateria e pasta são movimentados para A2
02:40	Documento de carro e chaveiro são movimentados para A3
02:45	Chaveiro é movimentado para A2
02:50	Carregador de bateria, pasta e chaveiro são movimentados para A4;
02:55	Documento do carro é movimentado para A1
03:00	Roubo do carregador de bateria, pasta e chaveiro
03:05	Carregador de bateria, pasta e chaveiro são movimentados para A3
03:10	pasta e chaveiro são movimentados para A1
03:15	Usuário sai com o documento do carro, chaveiro e pasta
03:20	Documento do carro é movimentado para A3; chaveiro é movimentado para A2
03:25	Carregador de bateria e documento do carro são movimentados para A1; pasta é movimentada para A3
03:30	Roubo do documento do carro, carregador de bateria, pasta e chaveiro
03:35	Documento do carro, carregador de bateria, pasta e chaveiro são movimentados para A4
03:40	Documento do carro, carregador de bateria são movimetnados para A1; pasta e chaveiro são movimentados para A2
03:45	Carregador de bateria é movimentado para A3; chaveiro é movimentado para A1
03:50	Documento do carro e chaveiro são movientados para A3; pasta é movimentada A4
03:55	Carregador de bteria, documento do carro e chaveiro são movimentados para A1; pasta é movimentada para A1

Movimentação dos objetos dentro de mochila.

Tabela I.2: Planejamento da simulação 2 da 4ª até a 5ª hora.

Horário	Movimentações
04:00	Documento do carro é movimentado para A2
04:05	Carregador de bateria é movimentado para A3
04:10	Pasta é movimentada para A4
04:15	Usuário sai com o documento do carro e carregador de bateria
04:20	Chaveiro é movimentdo para A4
04:25	Documento do carro é movimentado para A3
04:30	Roubo dos chaveiro e pasta
04:35	Carregador de bateria e documento do carro são movimentados para A4
04:40	Chaveiro e pasta são movimentados para A2
04:45	Carregador de bateria é movimentado para A2
04:50	Carregador de bateria, pasta e chaveiro são movimentados para A4
04:55	Documento do carro, carregador de bateria, pasta e chaveiro são movimentados para A1

Movimentação dos objetos dentro dos bolsos.

Tabela I.3: Planejamento da simulação 2 da 5ª até a 6ª hora.

Horário	Movimentações
05:00	Documento do carro é movimentado para A2
05:05	Carregador de bateria é movimentado para A3
05:10	Pasta é movimentada para A4
05:15	Usuário sai com o documento do carro e carregador de bateria
05:20	Chaveiro é movimentdo para A4
05:25	Documento do carro é movimentado para A3
05:30	Roubo dos chaveiro e pasta
05:35	Carregador de bateria e documento do carro são movimentados para A4
05:40	Chaveiro e pasta são movimentados para A2
05:45	Carregador de bateria é movimentado para A2
05:50	Carregador de bateria, pasta e chaveiro são movimentados para A4
05:55	Documento do carro, carregador de bateria, pasta e chaveiro são movimentados para A1

Movimentação dos objetos com os usuários em posições diferentes.

Tabela I.4: Planejamento da simulação 2 da 6ª até a 7ª hora.

Horário	Movimentações
06:00	Documento do carro é movimentado para A2 segurado no alto pelo usuário
06:05	Carregador de bateria é movimentado para A3 com o usuário agachado
06:10	Pasta é movimentada para A4 com o usuário ratejando
06:15	Usuário sai com o documento do carro e carregador de bateria pulando
06:20	Chaveiro é movimentdo para A4 dentro das mãos do usuário
06:25	Documento do carro é movimentado para A3 agachado
06:30	Roubo dos chaveiro e pasta carregados no alto pelo ladrão
06:35	Carregador de bateria e documento do carro são movimentados para A4 com um empilhado no outro
06:40	Chaveiro e pasta são movimentados para A2 com o usuário ratejando
06:45	Carregador de bateria é movimentado para A2 com o usuário correndo
06:50	Carregador de bateria, pasta e chaveiro são movimentados para A4 com o usuário pulando
06:55	Documento do carro, carregador de bateria, pasta e chaveiro são movimentados para A1 segurado no alto pelo usuário

II. PROGRAMAS DESENVOLVIDOS

Foram desenvolvidos os seguintes códigos nesse trabalho:

II.1 Aplicativo para Android:

II.1.1 Objetos

```
package erik.wh2;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;
public class Objetos {
public static final String KEY_ROWID="_rowid";
public static final String KEY_ID="objeto_id";
public static final String KEY_NAME="objeto_nome";
public static final String KEY_CRONOGRAMA="objeto_cron";
public static final String KEY_OP="objeto_opcao";
private static final String DATABASE_NAME="Objetosdb";
private static final String DATABASE_TABLE="objetosTABLE";
private static final int DATABASE_VERSION=2;
private DbHelper ourHelper;
private final Context ourContext;
private SQLiteDatabase ourDatabase;
private static class DbHelper extends SQLiteOpenHelper{
public DbHelper(Context context) {
super(context, DATABASE_NAME, null, DATABASE_VERSION);
```

```
// TODO Auto-generated constructor stub
}
@Override
public void onCreate(SQLiteDatabase db) {
// TODO Auto-generated method stub
db.execSQL("CREATE TABLE " + DATABASE_TABLE+ " ("+
KEY_ROWID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
KEY_ID + " TEXT NOT NULL, " +
KEY_NAME + " TEXT NOT NULL, " +
KEY_CRONOGRAMA + " TEXT NOT NULL, " +
KEY_OP + " TEXT NOT NULL);"
);
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
// TODO Auto-generated method stub
db.execSQL("DROP TABLE IF EXISTS " + DATABASE_TABLE);
onCreate(db);
}
}
public Objetos(Context c){
ourContext= c;
}
public Objetos open() throws SQLException{
ourHelper = new DbHelper(ourContext);
ourDatabase = ourHelper.getWritableDatabase();
return this;
}
public void close(){
ourHelper.close();
}
```

```
public long createEntry(String nome1, String id1) {
// TODO Auto-generated method stub
ContentValues cv= new ContentValues();
cv.put(KEY_ID, id1);
cv.put(KEY_NAME, nome1);
cv.put(KEY_CRONOGRAMA,"");
cv.put(KEY_OP,"00");
return ourDatabase.insert(DATABASE_TABLE, null, cv);
}
public String getData() {
// TODO Auto-generated method stub
String[] columns = new String[]{KEY_ROWID,KEY_ID,KEY_NAME,KEY_CRONOGRAMA,KEY_OP}
Cursor c = ourDatabase.query(DATABASE_TABLE, columns, null, null, null, null, null);
String result="";
for(c.moveToFirst(); !c.isAfterLast();c.moveToNext()){
result=result+c.getString(0)+" "+c.getString(1)+" "+
c.getString(2)+" "+ c.getString(3)+" "+
c.getString(4)+ "\n";
}
return result;
}
public void delete() {
// TODO Auto-generated method stub
ourDatabase.delete(DATABASE_TABLE, null, null);
}
public String getNames() {
// TODO Auto-generated method stub
String[] columns = new String[]{KEY_ROWID,KEY_ID,KEY_NAME,KEY_CRONOGRAMA,KEY_OP}
Cursor c = ourDatabase.query(DATABASE_TABLE, columns, null, null, null, null, null);
if(c!=null){
String result="";
```

```
for(c.moveToFirst(); !c.isAfterLast();c.moveToNext()){
result=result+c.getString(2)+",";
}
return result;
}else
return null;
}
public void alteraNome(String nomeAntigo, String idnome) {
// TODO Auto-generated method stub
ContentValues cvUpdate= new ContentValues();
cvUpdate.put(KEY_NAME, idnome);
ourDatabase.update(DATABASE_TABLE, cvUpdate, KEY_ROWID+"="+nomeAntigo, null);
}
public String getRowIDs() {
// TODO Auto-generated method stub
String[] columns = new String[]{KEY_ROWID,KEY_ID,KEY_NAME,KEY_CRONOGRAMA,KEY_OP}
Cursor c = ourDatabase.query(DATABASE_TABLE, columns, null, null, null, null, null);
if(c!=null){
String result="";
for(c.moveToFirst(); !c.isAfterLast();c.moveToNext()){
result=result+c.getString(0)+",";
}
return result;
}else
return null;
}
/*public void incluiCronograma(int i, String l_begin, String l_end_dur,String freq,String
RowId) {
// TODO Auto-generated method stub
String[] columns = new String[]{KEY_ROWID,KEY_ID,KEY_NAME,KEY_CRONOGRAMA,KEY_OP}
Cursor c = ourDatabase.query(DATABASE_TABLE, columns, KEY_ROWID+"="+RowId,
```

```
null, null, null, null);
    if(c!=null){
    c.moveToFirst();
    String cronoAntigo=c.getString(3);
    ContentValues cvUpdate= new ContentValues();
    String cronograma =i+":::"+l_begin+":::"+l_end_dur+":::"+freq+"!";
    cronograma=cronoAntigo+cronograma;
    cvUpdate.put(KEY_CRONOGRAMA, cronograma);
    ourDatabase.update(DATABASE_TABLE, cvUpdate, KEY_ROWID+"="+RowId, null);
    }
    }*/
    public String getCronograma(String rowId) {
    // TODO Auto-generated method stub
    String[] columns = new String[]{KEY_ROWID,KEY_ID,KEY_NAME,KEY_CRONOGRAMA,KEY_OP}
    Cursor c = ourDatabase.query(DATABASE_TABLE, columns, KEY_ROWID+"="+rowId,
null, null, null, null);
    if(c!=null){
    c.moveToFirst();
    String crono=c.getString(3);
    return crono;
    }else
    return null;
    }
    public void deleteCrono(String rowId) {
    // TODO Auto-generated method stub
    ContentValues cvUpdate= new ContentValues();
    String cronograma="";
    cvUpdate.put(KEY_CRONOGRAMA, cronograma);
    ourDatabase.update(DATABASE_TABLE, cvUpdate, KEY_ROWID+"="+rowId, null);
    }
    public void incluiCronograma(String rowId, String data) {
```

```
// TODO Auto-generated method stub
ContentValues cvUpdate= new ContentValues();
cvUpdate.put(KEY_CRONOGRAMA, data);
ourDatabase.update(DATABASE_TABLE, cvUpdate, KEY_ROWID+"="+rowId, null);
}
public String getOp(String rowId) {
// TODO Auto-generated method stub
String[] columns = new String[]{KEY_ROWID,KEY_ID,KEY_NAME,KEY_CRONOGRAMA,KEY_OP}
Cursor c = ourDatabase.query(DATABASE_TABLE, columns, KEY_ROWID+"="+rowId,
null, null, null, null);
if(c!=null){
c.moveToFirst();
String opcao=c.getString(4);
return opcao;
}else
return null;
}
public void incluiOp(String rowId, String opcoes) {
// TODO Auto-generated method stub
ContentValues cvUpdate= new ContentValues();
cvUpdate.put(KEY_OP, opcoes);
ourDatabase.update(DATABASE_TABLE, cvUpdate, KEY_ROWID+"="+rowId, null);
}
public boolean idExiste(String socketData) {
// TODO Auto-generated method stub
String[] columns = new String[]{KEY_ROWID,KEY_ID,KEY_NAME,KEY_CRONOGRAMA,KEY_OP}
Cursor c = ourDatabase.query(DATABASE_TABLE, columns, null, null, null, null, null);
boolean existe;
if(c!=null){
existe=false;
for(c.moveToFirst(); !c.isAfterLast();c.moveToNext()){
```

```
if(socketData.equals(c.getString(1))){
existe=true;;
}
}
return existe;
}else
return false;
}
public String getRowID(String socketData) {
// TODO Auto-generated method stub
String[] columns = new String[]{KEY_ROWID,KEY_ID,KEY_NAME,KEY_CRONOGRAMA,KEY_OP}
Cursor c = ourDatabase.query(DATABASE_TABLE, columns, null, null, null, null, null);
if(c!=null){
String rowId="";
for(c.moveToFirst(); !c.isAfterLast();c.moveToNext()){
if(socketData.equals(c.getString(1))){
rowId=c.getString(0);
}
}
return rowId;
}else
return null;
}
public String getName(String rowId) {
// TODO Auto-generated method stub
String[] columns = new String[]{KEY_ROWID,KEY_ID,KEY_NAME,KEY_CRONOGRAMA,KEY_OP}
Cursor c = ourDatabase.query(DATABASE_TABLE, columns, KEY_ROWID+"="+rowId,
null, null, null, null);
if(c!=null){
c.moveToFirst();
String nome=c.getString(2);
```

```
return nome;
}else
return null;
}
public String getOpTodas() {
// TODO Auto-generated method stub
String[] columns = new String[]{KEY_ROWID,KEY_ID,KEY_NAME,KEY_CRONOGRAMA,KEY_OP}
Cursor c = ourDatabase.query(DATABASE_TABLE, columns, null, null, null, null, null);
if(c!=null){
String result="";
for(c.moveToFirst(); !c.isAfterLast();c.moveToNext()){
result=result+c.getString(4)+",";
}
return result;
}else
return null;
}
public String getCronogramaS() {
// TODO Auto-generated method stub
String[] columns = new String[]{KEY_ROWID,KEY_ID,KEY_NAME,KEY_CRONOGRAMA,KEY_OP}
Cursor c = ourDatabase.query(DATABASE_TABLE, columns, null, null, null, null, null);
if(c!=null){
String result="";
for(c.moveToFirst(); !c.isAfterLast();c.moveToNext()){
result=result+c.getString(3)+",";
}
return result;
}else
return null;
}
public void deleteRow(String rowId) {
```

```
// TODO Auto-generated method stub
ourDatabase.delete(DATABASE_TABLE, KEY_ROWID+"="+rowId, null);
}
}
```

II.1.2 Main Activity:

```
import java.io.IOException;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketTimeoutException;
import java.util.Calendar;
import java.util.List;
import java.util.Scanner;
import erik.wh2.R;
import erik.wh2.R.id;
import erik.wh2.R.layout;
import erik.wh2.R.menu;
import android.os.Bundle;
import android.os.Handler;
import android.app.Activity;
import android.app.ListActivity;
import android.app.PendingIntent;
import android.content.Intent;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentTransaction;
import android.support.v4.app.ListFragment;
import android.telephony.SmsManager;
import android.util.Log;
import android.view.KeyEvent;
```

```
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
public class MainActivity extends Activity {
public static final String TAG = "Connection";
public static final int TIMEOUT = 10;
Button mais;
String sObjetos [],rIds[],opTodas[],cronoS[],objetosComUsuario;
ListView listaObjetos;
//TextView status;
boolean usuario;
private String connectionStatus = null;
private String socketData = null;
private Handler mHandler,mHandler2 = null;
ServerSocket server = null;
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
usuario=false;
mais= (Button) findViewById(R.id.button1);
//status= (TextView) findViewById(R.id.tvStatus);
mais.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
```

```
// TODO Auto-generated method stub
new Thread(initializeConnection).start();
String msg = "Attempting to connect...";
Toast.makeText(MainActivity.this, msg, msg.length()).show();
}
});
listaObjetos = (ListView) findViewById(R.id.lvObjetos);
mHandler = new Handler();
mHandler2=new Handler();
objetosComUsuario="";
}
@Override
protected void onResume() {
// TODO Auto-generated method stub
super.onResume();
Objetos ob =new Objetos(this);
sObjetos=null;
ob.open();
String nomes = ob.getNames();
String rowIDs= ob.getRowIDs();
String opTodas2=ob.getOpTodas();
String cadaCrono=ob.getCronogramaS();
ob.close();
sObjetos=nomes.split(",");
rIds=rowIDs.split(",");
opTodas=opTodas2.split(",");
cronoS=cadaCrono.split(",");
if (sObjetos==null){
sObjetos= new String []{"Objeto 1","Objeto 2","Objeto 3","Objeto 4"};
}
ArrayAdapter adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
```

```
sObjetos);
    listaObjetos.setAdapter(adapter);
    // Create a message handling object as an anonymous class.
    OnItemClickListener itemClicado = new OnItemClickListener() {
    public void onItemClick(AdapterView parent, View v, int position, long id) {
    // Do something in response to the click
    Bundle basket = new Bundle();
    basket.putString("sObjetos", sObjetos[position]);
    basket.putString("RowIds", rIds[position]);
    Intent a = new Intent (MainActivity.this, DefObjeto.class);
    a.putExtras(basket);
    startActivity(a);
    }
    };
    listaObjetos.setOnItemClickListener(itemClicado);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.activity_main, menu);
    return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
    // TODO Auto-generated method stub
    switch(item.getItemId()){
    case R.id.menu_settings:
    Intent i = new Intent("erik.wh2.SQLVIEW");
    startActivity(i);
    break;
    }
```

```
return false;
}
private Runnable initializeConnection = new Thread() {
public void run() {
Socket client = null;
// initialize server socket
try {
server = new ServerSocket(38300);
server.setSoTimeout(TIMEOUT * 1000);
// attempt to accept a connection
client = server.accept();
Globals.socketIn = new Scanner(client.getInputStream());
Globals.socketOut = new PrintWriter(client.getOutputStream(),
true);
// Globals.socketIn.
} catch (SocketTimeoutException e) {
// print out TIMEOUT
connectionStatus = "Connection has timed out! Please try again";
mHandler.post(showConnectionStatus);
} catch (IOException e) {
Log.e(TAG, "" + e);
} finally {
// close the server socket
try {
if (server != null)
server.close();
} catch (IOException ec) {
Log.e(TAG, "Cannot close server socket" + ec);
}
}
if (client != null) {
```

```
Globals.connected = true;
// print out success
connectionStatus = "Connection was succesful!";
Log.d(TAG, "connected!");
mHandler.post(showConnectionStatus);
//status.setText(connectionStatus);
while (Globals.socketIn.hasNext()) {
socketData = Globals.socketIn.next();
if(socketData.equals("ad1b030144b217956a00003e")){
Log.d(TAG, "Usuario!");
Thread a =new Thread(socketStatus2);
a.start();
}
else{
Log.d(TAG, "Objeto");
Thread b =new Thread(socketStatus);
b.start();
}
}
// startActivity(i);
}
}
};
private Runnable showConnectionStatus = new Runnable() {
public void run() {
Toast.makeText(getBaseContext(), connectionStatus,
Toast.LENGTH_SHORT).show();
}
};
private Runnable socketStatus2= new Thread(){
public void run() {
```

```
Log.d(TAG, "Usuario2!");
usuario=true;
try {
Thread.sleep(5000);
Thread.yield();
} catch (InterruptedException e) {
// TODO Auto-generated catch block
Log.e(TAG, "" + e);
}finally{
String lista = listaObjetosDoDia();
Log.d(TAG, "chegou"+lista);
if (!lista.equals("")){
String mensagemOb="O(s) objeto(s) abaixo nao estao com o usuario: \n";
String [] lista2=lista.split(":::");
boolean faltou=false;
String[] obComUsu=objetosComUsuario.split(":::");
for(int x=0;x<lista2.length;x++){
for(int y=0;y<obComUsu.length;y++){
if(lista2[x].equals(obComUsu[y])){
lista2[x]="";
}
}
if(!lista2[x].equals("")){
Objetos ob1=new Objetos(MainActivity.this);
ob1.open();
String nomeA=ob1.getName(lista2[x]);
ob1.close();
mensagemOb+="- "+nomeA+";\n";
faltou=true;
}
}
```

```
if(faltou){
sendSMS(mensagemOb);
Log.d(TAG, ""+mensagemOb);
}
}
objetosComUsuario="";
usuario=false;
}
}
private String listaObjetosDoDia() {
// TODO Auto-generated method stub
String resultado="";
for(int n=0;n<opTodas.length;n++){
if(opTodas[n].charAt(0)=='1'){
if(!cronoS[n].equals("")){
String[] cronogramaAlarme=cronoS[n].split(":::");
Calendar c = Calendar.getInstance();
if(cronogramaAlarme[0].equals("0")){
int dia = c.get(Calendar.DAY_OF_WEEK);
if(cronogramaAlarme[1].charAt(dia-1)=='1'){
resultado+=rIds[n]+":::";
}
}
if(cronogramaAlarme[0].equals("1")){
String ano =""+ c.get(Calendar.YEAR);
int mes1=c.get(Calendar.MONTH)+1;
String mes =""+mes1;
String dia =""+c.get(Calendar.DAY_OF_MONTH);
String[] data2=cronogramaAlarme[1].split("/");
Log.d(TAG,dia+data2[0]+"!"+mes+data2[1]+"!"+ano+data2[2]);
if(data2[0].equals(dia)&&data2[1].equals(mes)&&data2[2].equals(ano)){
```

```
resultado+=rIds[n]+":::";
}
}
}
}
}
return resultado;
}
};
private Runnable socketStatus = new Thread() {
public void run() {
Objetos ob=new Objetos(MainActivity.this);
ob.open();
if(ob.idExiste(socketData)){
String rowId =ob.getRowID(socketData);
String opcao=ob.getOp(rowId);
if (Globals.socketOut != null) {
if(usuario)
Globals.socketOut.println("Usuario");
else
Globals.socketOut.println(ob.getName(rowId));
Globals.socketOut.flush();
}
if(opcao.equals("00")||opcao.equals("")){
//nada
}else{
if(usuario){
Log.d(TAG, "FUNCIONOU");
objetosComUsuario+=rowId+":::";
}else{
int n=0;
```

```
while(usuario==false&&n<5){
try {
Thread.sleep(1000);
Thread.yield();
} catch (InterruptedException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}finally{
n++;
}
}
if(usuario){
Log.d(TAG, "FUNCIONOU2");
objetosComUsuario+=rowId+":::";
}else{
Log.d(TAG, "Ultimo");
if(opcao.charAt(1)=='1'){
Log.d(TAG, "Saida desautorizada do objeto "+ob.getName(rowId));
sendSMS("Saida desautorizada do objeto "+ob.getName(rowId));
}
}
}
}
/*
if(opcao.equals("00"))
{
//nada
}else{
if(opcao.charAt(1)=='0'){
//nada;
}else{
```

```
}
}*/
}
else{
ob.createEntry(socketData, socketData);
}
ob.close();
}
};
private void sendSMS(String mensagem) {
// TODO Auto-generated method stub
PendingIntent pi = PendingIntent.getActivity(MainActivity.this, 0,
new Intent(), 0);
SmsManager sms = SmsManager.getDefault();
sms.sendTextMessage("83130404", null, mensagem, pi, null);
}
public static class Globals {
public static boolean connected;
public static Scanner socketIn;
public static PrintWriter socketOut;
}
}
```

II.1.3 DefObjeto

```
package erik.wh2;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import erik.wh2.R;
import erik.wh2.R.id;
import erik.wh2.R.layout;
```

```
import android.app.Activity;
import android.app.Dialog;
import android.content.ContentUris;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.text.format.Time;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
public class DefObjeto extends Activity{
Button mudarnome,cronograma,alarme,deleta;
TextView nome,crono;
String idnome,rowId;
int n;
boolean alterouCrono=true;
@Override
protected void onCreate(Bundle savedInstanceState) {
// TODO Auto-generated method stub
super.onCreate(savedInstanceState);
alterouCrono=false;
setContentView(R.layout.def_objeto);
mudarnome = (Button) findViewById(R.id.btAlterarNome);
cronograma = (Button) findViewById(R.id.btIncluirCronograma);
alarme = (Button) findViewById(R.id.btAlarme1);
deleta = (Button) findViewById(R.id.btDeleta);
nome = (TextView) findViewById(R.id.tvNome);
crono = (TextView) findViewById(R.id.tvCronoObjeto);
Bundle gotBasket = getIntent().getExtras();
```

```
idnome=gotBasket.getString("sObjetos");
rowId=gotBasket.getString("RowIds");
mudarnome.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
// TODO Auto-generated method stub
Intent i = new Intent(DefObjeto.this,AlteraNome.class);
startActivityForResult(i,0);
}
});
cronograma.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
// TODO Auto-generated method stub
/*Intent intent = new Intent(Intent.ACTION_EDIT);
intent.setType("vnd.android.cursor.item/event");
intent.putExtra("title", idnome);
intent.putExtra("has_alarm", 0);
intent.putExtra("description", "NaoAltere");
intent.putExtra("name", "WhereHouse");
startActivity(intent);
alterouCrono=true;*/
Bundle basket = new Bundle();
basket.putString("Nome", idnome);
basket.putString("RowId", rowId);
Intent crono=new Intent("erik.wh2.DEFCRONO");
crono.putExtras(basket);
startActivity(crono);
}
});
alarme.setOnClickListener(new View.OnClickListener() {
```

```
@Override
public void onClick(View v) {
// TODO Auto-generated method stub
Bundle basket = new Bundle();
basket.putString("Nome", idnome);
basket.putString("RowId", rowId);
Intent alarm=new Intent("erik.wh2.DEFALARME");
alarm.putExtras(basket);
startActivity(alarm);
}
});
deleta.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
// TODO Auto-generated method stub
Objetos delOb= new Objetos(DefObjeto.this);
delOb.open();
delOb.deleteRow(rowId);
delOb.close();
finish();
}
});
}
//private String m_selectedCalendarId = "0";
private static final String DATE_TIME_FORMAT = "dd MMM yyyy, HH:mm:ss";
public static String getDateTimeStr(int p_delay_min) {
Calendar cal = Calendar.getInstance();
SimpleDateFormat sdf = new SimpleDateFormat(DATE_TIME_FORMAT);
if (p_delay_min == 0) {
return sdf.format(cal.getTime());
} else {
```

```
Date l_time = cal.getTime();
l_time.setMinutes(l_time.getMinutes() + p_delay_min);
return sdf.format(l_time);
}
}
public static String getDateTimeStr(String p_time_in_millis) {
SimpleDateFormat sdf = new SimpleDateFormat(DATE_TIME_FORMAT);
Date l_time = new Date(Long.parseLong(p_time_in_millis));
return sdf.format(l_time);
}
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
// TODO Auto-generated method stub
super.onActivityResult(requestCode, resultCode, data);
if (resultCode== RESULT_OK){
Bundle extras = data.getExtras();
idnome=extras.getString("alterou");
Objetos ob = new Objetos(DefObjeto.this);
ob.open();
ob.alteraNome(rowId,idnome);
ob.close();
}
}
@Override
protected void onResume() {
// TODO Auto-generated method stub
super.onResume();
Objetos ob = new Objetos(this);
String cronoFinal="";
nome.setText(idnome);
ob.open();
```

```
String cronoText=ob.getCronograma(rowId);
ob.close();
if (cronoText!=null){
String cronotext2[]=cronoText.split(":::");
if (cronotext2[0].equals("0")){
cronoFinal="Dias da semana: \n";
char c []= new char[7];
if(cronotext2[1].charAt(0)=='1')
cronoFinal+="-Domingo\n";
if(cronotext2[1].charAt(1)=='1')
cronoFinal+="-Segunda\n";
if(cronotext2[1].charAt(2)=='1')
cronoFinal+="-Ter
```

III. HISTÓRICO DO PROGRAMA SUPERVISÓRIO DURANTE A SIMULAÇÃO 2

III.1 Documento do carro

- Mon Apr 15 23:00:45 BRT 2013: A1
- Mon Apr 15 23:00:48 BRT 2013: A3
- Mon Apr 15 23:01:07 BRT 2013: A4 (Posição final - errada - faltou leitura da antena 2)
- Mon Apr 15 23:20:49 BRT 2013: A2
- Mon Apr 15 23:20:53 BRT 2013: A3 (Posição final - certa)
- Mon Apr 15 23:40:33 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 00:15:02 BRT 2013: A3
- Mon Apr 16 00:15:08 BRT 2013: A0 (Posição final - certa)
- Mon Apr 16 00:20:15 BRT 2013: A3 (Posição final - certa)
- Mon Apr 16 00:40:01 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 01:01:03 BRT 2013: A3
- Mon Apr 16 01:01:08 BRT 2013: A0 (Posição final - certa)
- Mon Apr 16 01:19:56 BRT 2013: A3 (Posição final - certa)
- Mon Apr 16 01:25:07 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 01:29:52 BRT 2013: A3
- Mon Apr 16 01:30:01 BRT 2013: A0 (Posição final - certa)
- Mon Apr 16 01:40:23 BRT 2013: A3 (Posição final - certa)
- Mon Apr 16 01:55:07 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 02:15:26 BRT 2013: A3 (Posição final - errada - faltou leitura da antena 4)
- Mon Apr 16 02:20:02 BRT 2013: A0 (Posição final - errada)
- Mon Apr 16 02:30:36 BRT 2013: A3
- Mon Apr 16 02:30:40 BRT 2013: A0 (Posição final - certa)

- Mon Apr 16 02:35:06 BRT 2013: A3
- Mon Apr 16 02:35:13 BRT 2013: A4 (Posição final - certa)
- Mon Apr 16 02:40:09 BRT 2013: A3
- Mon Apr 16 02:40:11 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 02:50:01 BRT 2013: A3 (Posição final - certa)
- Mon Apr 16 02:54:58 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 03:00:26 BRT 2013: A3
- Mon Apr 16 03:00:29 BRT 2013: A4(posição final - errada - ausência de leitura da antena 3)
- Mon Apr 16 03:15:49 BRT 2013: A3(posição final - errada - ausência de leitura da antena 2 e 3)
- Mon Apr 16 04:00:11 BRT 2013: A1
- Mon Apr 16 04:00:14 BRT 2013: A3
- Mon Apr 16 04:00:14 BRT 2013: A4 (posição final - errada - devido ao erro da posição inicial)
- Mon Apr 16 04:25:26 BRT 2013: A3 (Posição final - certa)
- Mon Apr 16 04:25:28 BRT 2013: A4 (Posição final - certa)
- Mon Apr 16 04:15:12 BRT 2013: A2
- Mon Apr 16 04:15:15 BRT 2013: A3
- Mon Apr 16 04:15:19 BRT 2013: A0 (Posição final - certa)
- Mon Apr 16 04:25:00 BRT 2013: A3 (Posição final - certa)
- Mon Apr 16 04:36:00 BRT 2013: A4 (Posição final - certa)

III.2 Pasta

- Mon Apr 15 23:00:46 BRT 2013: A1
- Mon Apr 15 23:10:35 BRT 2013: A3
- Mon Apr 15 23:10:37 BRT 2013: A4 (Posição final - certa)
- Mon Apr 15 23:50:29 BRT 2013: A3 (Posição final - certa)
- Mon Apr 16 00:00:13 BRT 2013: A0 (Posição final - certa)
- Mon Apr 16 00:10:14 BRT 2013: A3

- Mon Apr 16 00:10:17 BRT 2013: A4 (Posição final - certa)
- Mon Apr 16 00:50:02 BRT 2013: A3
- Mon Apr 16 00:50:04 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 01:05:28 BRT 2013: A3 (Posição final - certa)
- Mon Apr 16 01:09:56 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 01:36:02 BRT 2013: A3
- Mon Apr 16 01:36:05 BRT 2013: A4 (Posição final - errada - faltou leitura da antena 1)
- Mon Apr 16 01:50:01 BRT 2013: A2 (Posição final - errada)
- Mon Apr 16 01:59:57 BRT 2013: A3
- Mon Apr 16 02:00:01 BRT 2013: A0 (Posição final - certa)
- Mon Apr 16 02:05:21 BRT 2013: A3 (Posição final - certa)
- Mon Apr 16 02:10:32 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 02:15:26 BRT 2013: A3
- Mon Apr 16 02:15:29 BRT 2013: A0 (Posição final - certa)
- Mon Apr 16 02:25:01 BRT 2013: A3 (Posição final - certa)
- Mon Apr 16 02:30:40 BRT 2013: A0 (Posição final - certa)
- Mon Apr 16 02:35:06 BRT 2013: A3
- Mon Apr 16 02:35:13 BRT 2013: A4 (Posição final - certa)
- Mon Apr 16 02:40:08 BRT 2013: A3 (Leitura indevida)
- Mon Apr 16 02:40:11 BRT 2013: A4 (Posição final - errada - devido a leitura indevida)
- Mon Apr 16 02:51:06 BRT 2013: A2 (Posição final - errada)
- Mon Apr 16 02:54:55 BRT 2013: A3
- Mon Apr 16 02:54:58 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 03:10:14 BRT 2013: A3(Posição final -errada - ausência da leitura da antena 3)
- Mon Apr 16 03:30:35 BRT 2013: A0 (Posição final - certa)
- Mon Apr 16 03:55:07 BRT 2013: A3 (Posição final - errada - ausência de várias leituras)
- Mon Apr 16 04:09:43 BRT 2013: A1
- Mon Apr 16 04:09:47 BRT 2013: A3(Posição final - errada - posição inicial incorreta)

- Mon Apr 16 04:40:15 BRT 2013: A0
- Mon Apr 16 04:40:18 BRT 2013: A3
- Mon Apr 16 04:40:22 BRT 2013: A4 (Posição final - errada - posição inicial incorreta)
- Mon Apr 16 04:50:29 BRT 2013: A2 (Posição final - errada - posição inicial incorreta)

III.3 Chaveiro

- Mon Apr 15 23:00:45 BRT 2013: A1
- Mon Apr 15 23:30:13 BRT 2013: A3
- Mon Apr 15 23:30:17 BRT 2013: A0 (Posição final - certa)
- Mon Apr 15 23:34:52 BRT 2013: A3 (Posição final - certa)
- Mon Apr 15 23:55:01 BRT 2013: A4 (Posição final - certa)
- Mon Apr 16 00:35:43 BRT 2013: A2 (Posição final - certa)
- Mon Apr 16 00:55:15 BRT 2013: A4
- Mon Apr 16 00:55:17 BRT 2013: A3 (Leitura indevida)
- Mon Apr 16 00:55:18 BRT 2013: A4
- Mon Apr 16 00:55:20 BRT 2013: A3 (Posição final - errada - devido a leitura indevida)
- Mon Apr 16 01:05:28 BRT 2013: A1 (Posição final - errada)
- Mon Apr 16 01:15:09 BRT 2013: A3 (Posição final - errada)
- Mon Apr 16 01:25:03 BRT 2013: A0 (posição final - errada - ausência de leitura da antena 1)
- Mon Apr 16 01:29:52 BRT 2013: A3
- Mon Apr 16 01:30:01 BRT 2013: A0 (Posição final - certa)
- Mon Apr 16 01:40:23 BRT 2013: A3 (Posição final - certa)
- Mon Apr 16 01:44:57 BRT 2013: A4
- Mon Apr 16 01:45:00 BRT 2013: A2 (Posição final - certa)
- Mon Apr 16 01:50:01 BRT 2013: A4 (Posição final - certa)
- Mon Apr 16 01:59:57 BRT 2013: A3
- Mon Apr 16 02:00:01 BRT 2013: A0 (Posição final - certa)

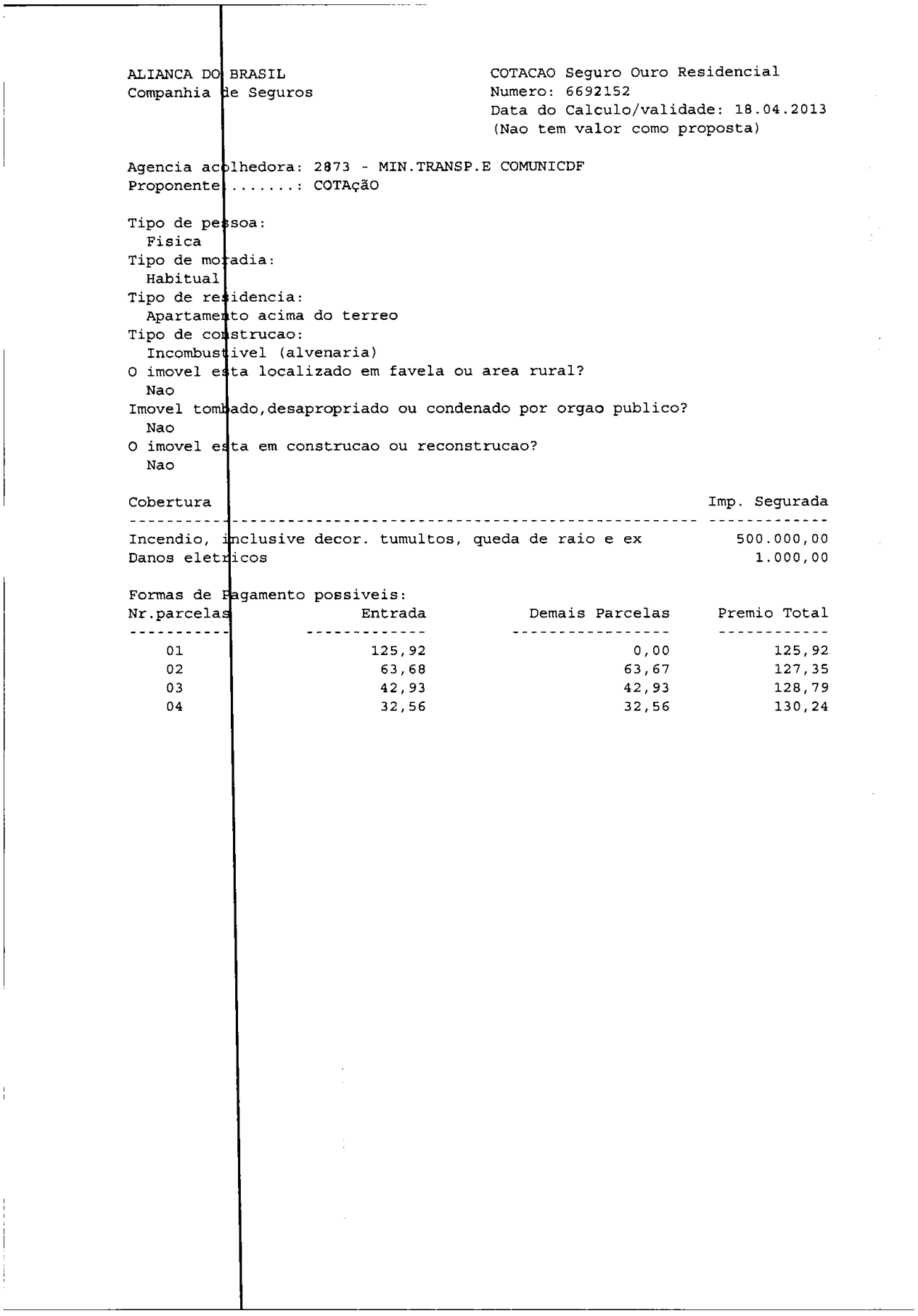
- Mon Apr 16 02:05:21 BRT 2013: A3 (Posição final - certa)
- Mon Apr 16 02:10:32 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 02:15:26 BRT 2013: A3 (Posição final - errada - ausência de leitura da antena 4)
- Mon Apr 16 02:20:01 BRT 2013: A4
- Mon Apr 16 02:20:04 BRT 2013: A3
- Mon Apr 16 02:20:07 BRT 2013: A4 (Posição final - errada)
- Mon Apr 16 02:30:36 BRT 2013: A3
- Mon Apr 16 02:30:40 BRT 2013: A0 (Posição final - certa)
- Mon Apr 16 02:35:06 BRT 2013: A3
- Mon Apr 16 02:35:13 BRT 2013: A4 (Posição final - certa)
- Mon Apr 16 02:40:11 BRT 2013: A2 (Posição final - certa)
- Mon Apr 16 02:45:01 BRT 2013: A4
- Mon Apr 16 02:45:03 BRT 2013: A3
- Mon Apr 16 02:45:07 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 02:50:01 BRT 2013: A3 (Posição final - certa)
- Mon Apr 16 02:54:53 BRT 2013: A4 (leitura indevida))
- Mon Apr 16 02:54:58 BRT 2013: A3 (Posição final - errada - devido a leitura indevida)
- Mon Apr 16 03:20:03 BRT 2013: A4 (Posição final - certa - ausência da leitura da antena 1)
- Mon Apr 16 03:39:48 BRT 2013: A3
- Mon Apr 16 03:39:53 BRT 2013: A4 (Posição final - errada - ausência de leitura da antena 3)
- Mon Apr 16 03:55:07 BRT 2013: A3 (Posição final - errada - ausência de várias leituras)
- Mon Apr 16 04:30:48 BRT 2013: A4
- Mon Apr 16 04:30:48 BRT 2013: A3 (Posição final - errada - posição inicial incorreta)
- Mon Apr 16 04:50:01 BRT 2013: A4 (Posição final - certa)
- Mon Apr 16 04:55:13 BRT 2013: A3
- Mon Apr 16 04:55:17 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 04:50:29 BRT 2013: A4 (Posição final - certa)

III.4 Carregador de bateria

- Mon Apr 15 23:00:46 BRT 2013: A1
- Mon Apr 15 23:05:12 BRT 2013: A3 (Posição final - certa)
- Mon Apr 15 23:44:59 BRT 2013: A4 (Posição errada - ausêcia de leitura da antena 3)
- Mon Apr 16 00:05:38 BRT 2013: A2
- Mon Apr 16 00:05:41 BRT 2013: A3
- Mon Apr 16 00:05:43 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 00:25:01 BRT 2013: A3
- Mon Apr 16 00:25:04 BRT 2013: A4
- Mon Apr 16 00:25:07 BRT 2013: A2 (Posição final - certa)
- Mon Apr 16 00:30:41 BRT 2013: A4
- Mon Apr 16 00:30:43 BRT 2013: A3
- Mon Apr 16 00:30:46 BRT 2013: A0 (Posição final - certa)
- Mon Apr 16 00:44:59 BRT 2013: A3 (Posição final - certa)
- Mon Apr 16 01:09:56 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 01:36:02 BRT 2013: A3
- Mon Apr 16 01:36:05 BRT 2013: A4 (Posição final - errada - faltou leitura da antena 1)
- Mon Apr 16 01:50:01 BRT 2013: A2 (Posição final - errada)
- Mon Apr 16 01:59:57 BRT 2013: A3
- Mon Apr 16 02:00:01 BRT 2013: A0 (Posição final - certa)
- Mon Apr 16 02:05:21 BRT 2013: A3 (Posição final - certa)
- Mon Apr 16 02:25:02 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 02:30:36 BRT 2013: A3
- Mon Apr 16 02:30:40 BRT 2013: A0 (Posição final - certa)
- Mon Apr 16 02:35:06 BRT 2013: A3
- Mon Apr 16 02:35:13 BRT 2013: A4 (Posição final - certa)
- Mon Apr 16 02:40:09 BRT 2013: A3

- Mon Apr 16 02:40:11 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 02:45:38 BRT 2013: A3 (Posição final - certa)
- Mon Apr 16 02:54:58 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 03:45:02 BRT 2013: A4 (Posição final - errada - ausência de várias leituras)
- Mon Apr 16 03:55:07 BRT 2013: A3 (Posição final - errada - ausência de várias leituras)
- Mon Apr 16 04:14:51 BRT 2013: A0 (Posição final - certa)
- Mon Apr 16 04:50:01 BRT 2013: A4 (Posição final - certa)
- Mon Apr 16 04:55:13 BRT 2013: A3
- Mon Apr 16 04:55:17 BRT 2013: A1 (Posição final - certa)
- Mon Apr 16 05:05:38 BRT 2013: A3 (Posição final - certa)
- Mon Apr 16 04:15:19 BRT 2013: A0 (Posição final - certa)
- Mon Apr 16 04:35:58 BRT 2013: A3
- Mon Apr 16 04:36:00 BRT 2013: A4 (Posição final - certa)
- Mon Apr 16 04:45:31 BRT 2013: A2 (Posição final - certa)
- Mon Apr 16 04:50:29 BRT 2013: A4 (Posição final - certa)

IV. COTAÇÃO DE SEGURO RESIDENCIAL COM O BANCO DO BRASIL



ALIANCA DO BRASIL
Companhia de Seguros

COTACAO Seguro Ouro Residencial
Numero: 6692152
Data do Calculo/validade: 18.04.2013
(Nao tem valor como proposta)

Agencia acolhedora: 2873 - MIN.TRANSP.E COMUNICDF
Proponente.....: COTAçãO

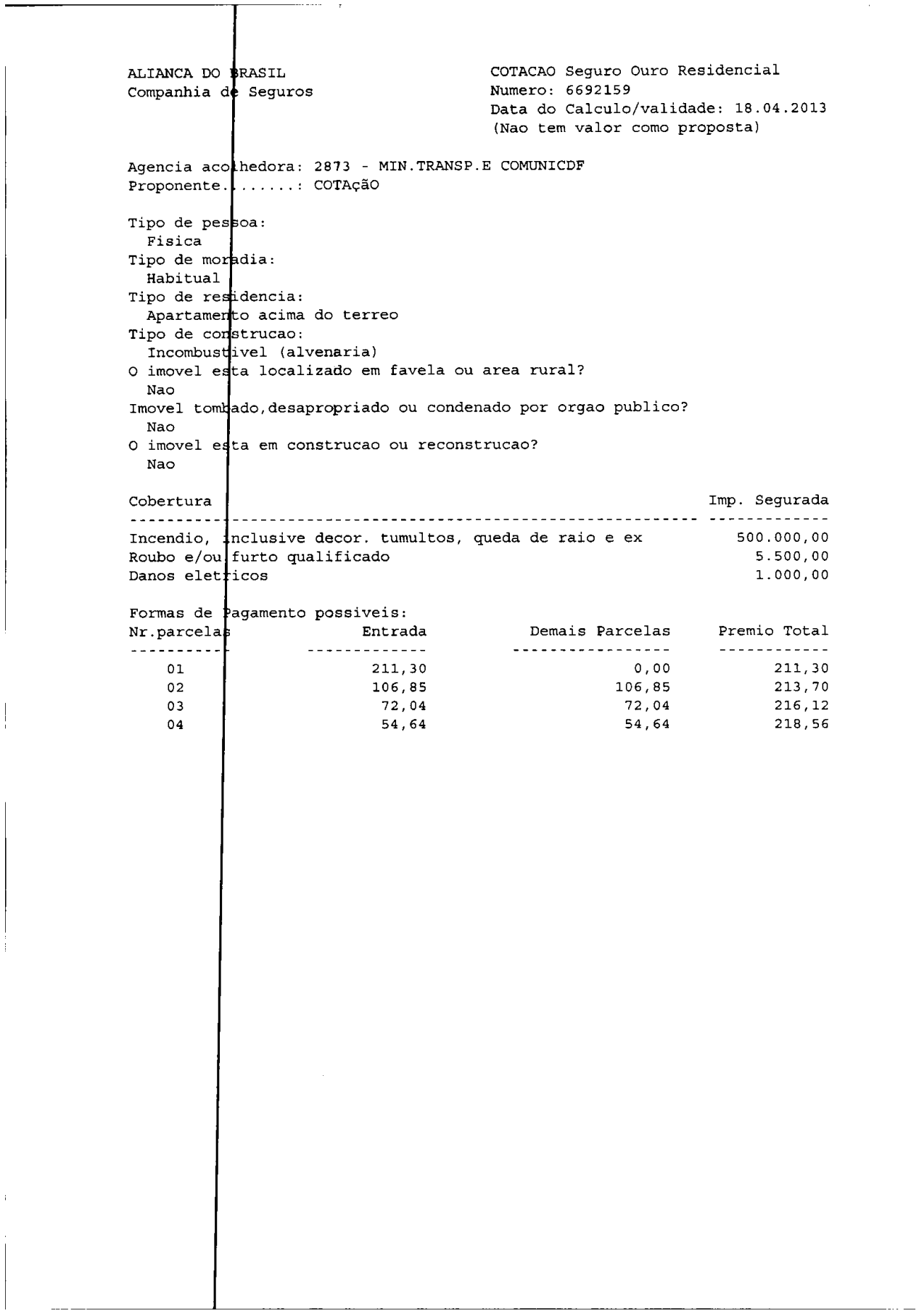
Tipo de pessoa:
Fisica
Tipo de moradia:
Habitual
Tipo de residencia:
Apartamento acima do terreo
Tipo de construcao:
Incombustivel (alvenaria)
O imovel esta localizado em favela ou area rural?
Nao
Imovel tombado,desapropriado ou condenado por orgao publico?
Nao
O imovel esta em construcao ou reconstrucao?
Nao

Cobertura	Imp. Segurada
Incendio, inclusive decor. tumultos, queda de raio e ex	500.000,00
Danos eletricos	1.000,00

Formas de Pagamento possiveis:

Nr.parcelas	Entrada	Demais Parcelas	Premio Total
01	125,92	0,00	125,92
02	63,68	63,67	127,35
03	42,93	42,93	128,79
04	32,56	32,56	130,24

Figura IV.1: Cotação de seguro residencial sem cobertura de roubo



ALIANCA DO BRASIL
Companhia de Seguros

COTACAO Seguro Ouro Residencial
Numero: 6692159
Data do Calculo/validade: 18.04.2013
(Nao tem valor como proposta)

Agencia acolhedora: 2873 - MIN.TRANSP.E COMUNICDF
Proponente.....: COTAçãO

Tipo de pessoa:
Fisica
Tipo de moradia:
Habitual
Tipo de residencia:
Apartamento acima do terreo
Tipo de construcao:
Incombustivel (alvenaria)
O imovel esta localizado em favela ou area rural?
Nao
Imovel tombado,desapropriado ou condenado por orgao publico?
Nao
O imovel esta em construcao ou reconstrucao?
Nao

Cobertura	Imp. Segurada
Incendio, inclusive decor. tumultos, queda de raio e ex	500.000,00
Roubo e/ou furto qualificado	5.500,00
Danos eletricos	1.000,00

Formas de Pagamento possiveis:

Nr.parcelas	Entrada	Demais Parcelas	Premio Total
01	211,30	0,00	211,30
02	106,85	106,85	213,70
03	72,04	72,04	216,12
04	54,64	54,64	218,56

Figura IV.2: Cotação de seguro residencial com cobertura de roubo