DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Sistema de segurança residencial integrado com aplicativo para smartphone.

Relatório submetido à Universidade Federal de Santa Catarina como requisito para a aprovação da disciplina:

DAS 5511: Projeto de Fim de Curso

Luan César Souza Volpato

Florianópolis, Julho de 2012

Sistema de segurança residencial integrado com aplicativo para smartphone.

Luan César Souza Volpato

Orientadores:

Cristiano Studzinski Souza / Eng.

_____
Assinatura do Orientador

Prof. *Rômulo Silva de Oliveira*

_____
Assinatura do Orientador

Este relatório foi julgado no contexto da disciplina
DAS 5501: Estágio e Controle e Automação Industrial
e aprovado na sua forma final pelo
Curso de Engenharia de Controle e Automação

Agradecimentos

Agradeço primeiramente aos meus pais por todo apoio que, mesmo a distância, foram imprescindíveis para que eu alcançasse meus objetivos.

Agradeço ao professor Rômulo da Silva Oliveira que foi um ótimo professor na graduação e sempre foi atencioso durante a orientação deste trabalho.

Agradeço ao Cristiano Studzinsk e ao Cleber Amaral pela oportunidade que me deram na Automatiza e pelas orientações nas atividades dentro da empresa. Agradeço também toda a equipe de desenvolvimento da empresa que sempre se mostrou disponível a ajudar na solução dos problemas.

Agradeço a todos os amigos que fiz durante a faculdade e com certeza levarei comigo para o resto da vida.

Agradeço a minha namorada pelo apoio, carinho e por deixar os meus dias mais felizes.

Por fim, agradeço a DEUS por me dar tudo o que tenho.

Resumo

Um projeto de um sistema de segurança para ambiente residencial está sendo desenvolvido na empresa Automatiza. Esse projeto tem como objetivo oferecer uma solução de segurança simples para casas e apartamentos integrada com um aplicativo para smartphones. Essa solução conta com um sistema de controle de acesso de uma porta, de um portão eletrônico, um alarme e um botão de pânico.

Este documento descreve as atividades deste projeto que foi desenvolvido como Projeto do Fim de Curso, disciplina do curso de Engenharia de Controle e Automação da UFSC.

Uma das atividades deste trabalho é o desenvolvimento de um aplicativo para smartphone. O aplicativo foi desenvolvido para sistema operacional Android. Neste documento são mostrados os detalhes do desenvolvimento desse aplicativo.

Outra atividade deste trabalho foi o desenvolvimento de um firmware para uma placa controladora. Esse firmware é responsável por controlar os dispositivos do sistema e responder as requisições do aplicativo. Aqui são mostrados os detalhes de desenvolvimento e implementação.

Por fim, são mostrados os resultados obtidos após o desenvolvimento e testes.

Abstract

A design of a security system for residential environment is being developed in the Automatiza company. This project aims to provide a simple security solution for homes and apartments integrated with an application for smartphones. This solution has the access control of a door, an electronic gate, an alarm and a panic button.

This document describes the activities developed in this project. This work was developed as the Ending Course Project in UFSC's Control and Automation Engineering College.

One of the activities of this work is to develop an application for smartphones. The application was developed for Android operational system. This document shows the details of the development of this application.

Another activity of this work was to develop a firmware to a controller board. This firmware is responsible for controlling the devices in the system and to answer the requests of the application. Here are shown the details of development and implementation.

Finally, we present the results obtained after the development and testing.

Sumário

Agradecimentos 3
Resumo 4
Abstract 5
Sumário 6
Lista de Figuras 9
Lista de Tabelas 11
Capítulo 1: Introdução 12
1.1: Contextualização 12
1.2: A empresa Automatiza 13
1.3: Objetivos 13
1.3.1: Desenvolvimento da central de controle e acionamento 14
1.3.2: Desenvolvimento do aplicativo para smartphone 14
1.4: Justificativa 14
1.4.1: Smartphone 15
1.4.2: Segurança residencial 16
1.5: Descrição 17
1.5.1: Contextualização no curso 17
1.5.2: Organização dos capítulos 17
Capítulo 2: O Projeto 19
2.1: Funcionalidades esperadas 19
2.2: Especificações técnicas 20
2.3: Módulos funcionais 23
Capítulo 3: Plataformas móveis 25
3.1: Escolha do sistema operacional 25

3.1.1: iOS 25
3.1.2: Android 26
3.1.3: Windows Phone 27
3.1.4: BlackBerry OS 27
3.1.5: Symbian 28
3.1.6: Quadro comparativo 29
3.1.7: Informações sobre o mercado dos smartphones 30
3.2: O sistema operacional Android 35
3.2.1: Características 35
3.2.2: Componentes de aplicação 37
Capítulo 4: O aplicativo 40
4.1: Metodologia 40
4.2: Ferramentas de gerenciamento de projeto 41
4.3: Diagrama de classes 41
4.3.1: Classe Conexao 42
4.3.2: Classe Login 43
4.3.3: Classe Alarme 43
4.3.4: Classe Porta 44
4.3.5: Classe PortaoEletronico 44
4.3.6: Classe Panico 44
4.4: Interface 45
4.4.1: Camada de apresentação 45
4.4.2: Camada de aplicação 46
4.5: Codificação e testes 48
Capítulo 5: Firmware 49
5.1: Hardware 49
5.2: FreeRTOS 50

5.3: Bibliotecas 51
5.4: Ferramentas 52
5.5: Implementação do código 52
5.5.1: Processo Roteador() 54
5.5.2: Processo Alarme() 55
5.5.3: Processo Panico() 57
5.5.4: Processo Porta() 58
5.5.5: Processo PortaoEletonico() 59
5.6: Módulo GSM 60
Capítulo 6: Integração 62
6.1: Protocolo TCP/IP 62
6.2: A interface de socket 64
6.3: A arquitetura cliente/servidor 66
6.4: Protocolo da aplicação 67
Capítulo 7: Testes e Resultados 72
7.1: Instalação 72
7.2: Avaliação do sistema 73
Capítulo 8: Conclusões e Perspectivas 75
Bibliografia: 77

Lista de Figuras

Figura 1 - Estimativa de vendas de dispositivos móveis inteligentes no mundo 15
Figura 2 - Esquema de interação do usuário com o sistema 20
Figura 3 - Esquema dos módulos do sistema 23
Figura 4 - Distribuição dos sistemas operacionais rodando nos dispositivos móveis 33
Figura 5- Arquitetura do sistema operacional Android 36
Figura 6 - Diagrama de classes do aplicativo 41
Figura 7 - Interfaces do aplicativo 46
Figura 8 - Diagrama de estados de navegação 46
Figura 9 - Placa NetControl 50
Figura 10 - Diagrama de funcionamento dos processos do firmware 53
Figura 11 - Fluxograma do processo de roteamento de mensagem 54
Figura 12 - Fluxograma do processo do alarme 56
Figura 13 - Fluxograma do processo do botão de pânico 57
Figura 14 - Fluxograma do processo da porta 58
Figura 15 - Fluxograma do processo do portão eletrônico 59
Figura 16 - As quatro camadas do modele TCP/IP e sua comparação com o modelo OSI 63
Figura 17 - Protocolos usados em cada camada do modelo TCP/IP 63
Figura 18 - Código da função de envio de mensagem do aplicativo 65
Figura 19 - Arquitetura cliente/servidor 67

Figura 20 - Estrutura da mensagem do protocolo de comunicação do aplicativo com a placa..... 68

Lista de Tabelas

Tabela 1 - Requisitos do projeto 21

Tabela 2 - Quadro comparativo das principais plataformas móveis do mercado 29

Tabela 3 - Vendas de dispositivos móveis no mundo no segundo trimestre de 2011 (milhares de unidades) 31

Tabela 4 - Vendas de smartphones no mundo por sistemas operacionais (milhares de unidades) 32

Tabela 5 - Projeção de vendas de dispositivos móveis de comunicação (milhares de unidades) 34

Tabela 6 - Caracteres de identificação de cada dispositivo do sistema 68

Tabela 7 - Requisições da porta 69

Tabela 8 - Requisições do portão eletrônico 69

Tabela 9 - Requisições do alarme 70

Tabela 10 - Requisições do botão de pânico 71

Capítulo 1: Introdução

1.1: Contextualização

A onda de furtos e assaltos sempre foi e será um motivo de preocupação para as famílias em suas residências, seja em casas ou apartamentos. A maioria das residências passa horas ou até dias na ausência de seus donos ficando expostas a ações de criminosos. Logo, dispor de um ambiente protegido e seguro, mesmo na ausência de seus donos, é imprescindível para o conforto das pessoas em seus lares.

A automação residencial, conhecida também como domótica, vem crescendo nos últimos anos em uma busca por conforto, segurança e até entretenimento nas casas das pessoas. A palavra domótica, derivada de *Domus* (casa) e Robótica (controle automatizado de um sistema), define o conceito da possibilidade de controlar de forma automática as casas, que vulgarmente costumam ser designadas de “casas inteligentes” [1].

Dentro do contexto de automação residencial, uma das áreas mais importantes tratadas é a questão da segurança nas casas e apartamentos. Muitas tecnologias integradas com sistemas computacionais vêm sendo desenvolvidas visando garantir maior segurança nos ambientes residenciais. Equipamentos como alarmes, câmeras de vigilância, interfones, portões eletrônicos, entre outros, estão cada vez mais presentes nas residências das pessoas. Ainda nessa linha, a integração desses equipamentos de segurança com as redes computacionais tem oferecido o conforto do acesso remoto aos dispositivos instalados nas residências a partir de qualquer ponto que se tenha acesso a internet.

Outro setor tecnológico que tem se desenvolvido de forma muito rápida é o de telefonia móvel. Os celulares deixaram de ser simples dispositivos de comunicação para se tornarem verdadeiros computadores de bolso. Os smartphones (celulares inteligentes), como são chamados os celulares mais modernos, rodam sistemas operacionais complexos e são equipados de muitos

recursos como acesso à internet, câmera, orientação de GPS, além de uma infinidade de aplicativos que podem ser instalados nos mesmos [2].

A possibilidade de controlar dispositivos de segurança de forma remota por meio de celulares somada a mobilidade que o mesmo pode oferecer ao usuário é a motivação do desenvolvimento deste trabalho. Este objetiva desenvolver um protótipo de um sistema de segurança para ambiente residencial onde toda sua interface com o usuário é feita por um aplicativo para smartphone.

1.2: A empresa Automatiza

A Automatiza é uma empresa que desenvolve tecnologia para sistemas de segurança e controle de acesso. Presente há mais de 14 anos no mercado, suas instalações físicas situam-se em Palhoça, na região da grande Florianópolis. Além de Santa Cataria, a empresa ainda está presente no estado do Paraná, Minas Gerais e São Paulo. Atualmente a empresa, além de oferecer seus produtos para todo o Brasil, fornece produtos para mais de dez países no mundo todo.

A Automatiza desenvolve e oferece soluções completas de segurança para empresas, condomínios, bancos, presídios e outras organizações. Dentre os produtos oferecidos podem-se citar: catracas, leitores de cartão de proximidade, fechaduras eletromagnéticas, leitores biométricos e softwares de supervisão dos sistemas de controle de acesso.

1.3: Objetivos

O projeto tem seus principais objetivos centrados no desenvolvimento de uma solução simples para automação da segurança de um ambiente residencial. A solução é dotada de um sistema de controle de acesso e um alarme. Toda a interface de interação do sistema com o usuário é feita por um aplicativo para smartphone.

Os objetivos específicos do projeto visam o desenvolvimento de uma central de controle e acionamento responsável pela interação com os dispositivos do sistema e também do desenvolvimento de um aplicativo para smartphone que faz o acionamento remoto do sistema.

1.3.1: Desenvolvimento da central de controle e acionamento

A central de controle e acionamento do sistema comunica-se com o aplicativo do smartphone e controla os dispositivos do sistema. A comunicação com o smartphone é feita via wireless dando ao usuário a mobilidade de poder interagir com o sistema de qualquer ponto de sua residência. Esta central recebe as requisições do aplicativo, realiza a leitura dos sinais dos sensores da porta e do alarme do sistema, toma as decisões de controle e faz o acionamento dos dispositivos ligados ao sistema.

1.3.2: Desenvolvimento do aplicativo para smartphone

O aplicativo para smartphone será a interface de comunicação do usuário com o sistema. Por meio do aplicativo poderá ser visualizado o estado do sistema e enviar comandos de ações. O objetivo da interface do aplicativo é ser simples e intuitiva para o usuário ao mesmo tempo que consiga atender aos requisitos do projeto.

1.4: Justificativa

A Automatiza é uma empresa que desenvolve soluções em controle de acesso que garantam segurança aos usuários de seus produtos. Os produtos que a empresa oferece foram desenvolvidos para atender ambientes, em quase toda sua maioria, comerciais e industriais.

Na intenção de conquistar clientes em ambientes residenciais a empresa busca uma solução simples e de baixo custo que possa atender casas e apartamentos. Nesse sentido, seguindo as tendências tecnológicas, a intenção

é que essa solução tenha interação com o usuário por meio de um smartphone, cada vez mais presente nas mãos das pessoas, provendo mobilidade e simplicidade no acesso ao sistema.

1.4.1: Smartphone

Não é difícil perceber o crescimento do uso de smartphone nos últimos anos. Segundo a IDC (*International Data Corporation*), empresa de consultoria de inteligência de mercado no segmento de tecnologia da informação, o número de dispositivos inteligentes (PCs, tablets e smartphones) vendidos no mundo foi superior a 916 milhões de unidades em 2011 [3]. Dessa quantidade, a maior fatia corresponde às vendas de smartphones e tablets como pode ser visto na Figura 1 - Estimativa de vendas de dispositivos móveis inteligentes no mundo logo abaixo.

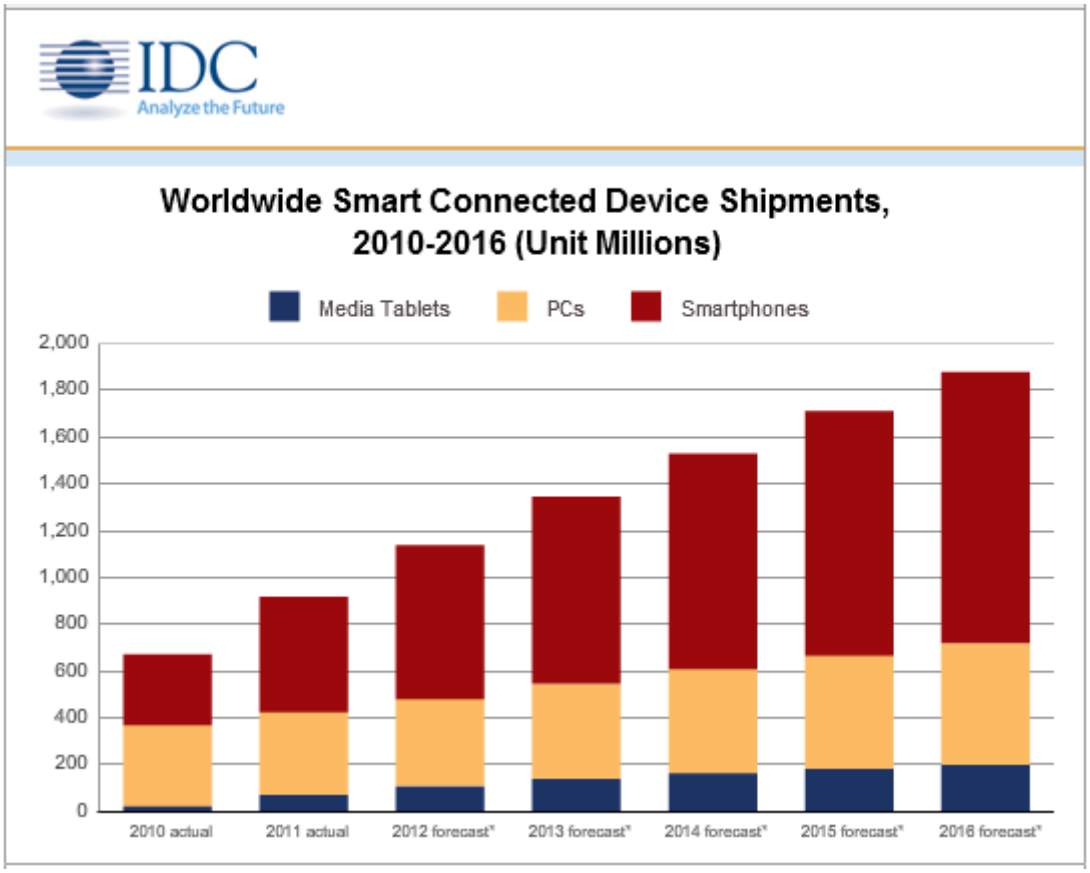

Figura 1 - Estimativa de vendas de dispositivos móveis inteligentes no mundo

Esse número tende a crescer ainda mais nos próximos anos devido ao avanço dos recursos disponíveis nos smartphones e pela queda no preço dos mesmos. Além das funções de um telefone móvel convencional, esses dispositivos inteligentes possuem conexão com rede de dados para acesso à internet, GPS, câmera, acelerômetro, giroscópio, editores de texto e uma variedade de aplicativos. Um dispositivo com todos esses recursos, que antes custava em torno de R$ 2000,00, hoje pode ser adquirido por menos de R$ 500,00. Esses fatores têm atraído muitos consumidores do mundo todo e, nesse cenário, o Brasil é um dos países mais promissores [4].

Outra característica interessante dos smartphones existentes no mercado é que eles possuem sistemas operacionais “abertos”, ou seja, é possível que qualquer pessoa possa desenvolver aplicativos para executar nesses dispositivos.

1.4.2: Segurança residencial

O interesse da empresa é criar um sistema de segurança para ambiente residencial com controle de acesso e alarme integrados com um aplicativo para smartphone. Pois foi percebida a tendência crescente do uso desses dispositivos para o controle de aparelhos domésticos.

Para isso, a proposta deste trabalho é o desenvolvimento de um firmware para a central de controle e alarme e um aplicativo para smartphone. É nestas atividades que este projeto está inserido.

O firmware da central de controle irá rodar na placa controladora NetControl da Automatiza e o aplicativo deverá ser compatível para a maioria das plataformas móveis existentes no mercado.

1.5: Descrição

1.5.1: Contextualização no curso

Dentro do currículo do curso de Engenharia de Controle e Automação pode-se observar algumas matérias que se relacionam, umas mais e outras menos intensamente, com o desenvolvimento deste trabalho. São elas:

- Sistemas Digitais / Microprocessadores / Informática Industrial 1 e 2: o projeto envolveu o desenvolvimento de firmware para hardware embarcado, o qual se valeu de conceitos abordados nessas disciplinas. Entre os conhecimentos adquiridos no curso usados no projeto pode-se citar a programação em linguagem C para microprocessadores, programação concorrente e protocolos de comunicação como TCP e UDP.
- Metodologia para Desenvolvimento de Sistemas / Redes de Computadores para Automação Industrial / Sistemas Distribuídos: o desenvolvimento de software de aplicativos para celular, ainda que não visto no curso, fez-se valer dos conhecimentos e métodos aprendidos nas disciplinas de informática do curso. Ainda vale citar os conhecimentos de redes de comunicação entre dispositivos microprocessados que foram importantes para a realização do trabalho.

1.5.2: Organização dos capítulos

A fim de informar o que será abordado em cada capítulo deste documento, este tópico traz uma breve descrição de como os capítulos estão organizados e quais assuntos tratados no mesmo.

O Capítulo 2: tem o objetivo de fazer uma apresentação geral do projeto, suas especificações, requisitos e como foram divididas as frentes de trabalho para o desenvolvimento do mesmo.

O Capítulo 3: apresenta alguns resultados de uma pesquisa sobre sistemas operacionais móveis que foi importante para a escolha das plataformas para as quais o projeto, assim que concluído, oferecerá suporte. Ainda mostra também algumas características do sistema operacional Android, o qual foi escolhido como a plataforma móvel do aplicativo desenvolvido neste trabalho.

O Capítulo 4: mostra detalhes do desenvolvimento do aplicativo para smartphone para sistema operacional Android, a metodologia usada, as ferramentas escolhidas e outros detalhes da implementação.

O Capítulo 5: trata do desenvolvimento do firmware, as características do hardware escolhido, do sistema operacional embarcado, a metodologia, as ferramentas usadas e como foi implementado.

O Capítulo 6: mostra como foi feita a integração do aplicativo com a placa controladora, a arquitetura usada e o protocolo de comunicação.

O capítulo 7 apresenta como foram feitos os testes do sistema e os resultados dos mesmos.

O capítulo 8 traz as conclusões do trabalho e apresenta algumas melhorias que ainda devem ser feitas.

Capítulo 2: O Projeto

O projeto desse sistema de segurança está relacionado a um produto novo da empresa. Até o início do desenvolvimento deste trabalho não havia definições sobre o escopo do projeto. Logo, as atividades deste trabalho abrangem desde o levantamento das definições das características do projeto, especificações de requisitos até a implementação.

Neste capítulo serão apresentadas as funcionalidades esperadas do sistema segurança, as especificações técnicas e como o trabalho foi dividido. Estas atividades estão inseridas na metodologia de como os requisitos de projeto foram levantados pela empresa. O estudos das alternativas de solução e implementação, que também foram levados em conta no levantamento desses requisitos. A solução será abordada nos próximos capítulos (3, 4 e 5).

2.1: Funcionalidades esperadas

Como já citado na seção 1.3, o projeto tem como objetivo desenvolver uma solução de sistema de segurança que seja simples e para um ambiente residencial.

A principal funcionalidade que o sistema deve ter é a interação com o usuário por meio de um aplicativo para smartphone. Espera-se que, por meio de uma interface gráfica simples e amigável para smartphone, o usuário possa enviar comandos de ações e receber informações do status do sistema. Essa integração do aplicativo com o sistema deve ser feita através de comunicação sem fio, pois garantir a mobilidade do usuário é uma característica desejada no produto.

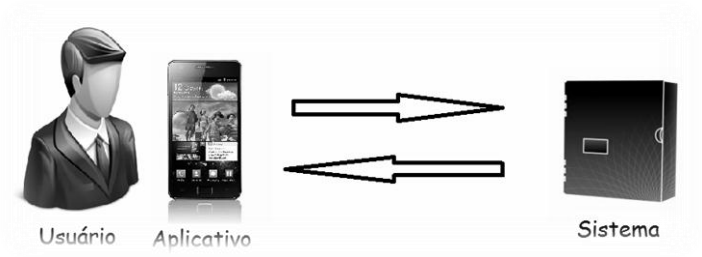

Figura 2 - Esquema de interação do usuário com o sistema

A Figura 2 - Esquema de interação do usuário com o sistema mostra um esquema de interação do usuário com o sistema. Por meio de uma rede sem fio, o aplicativo troca informações com o sistema e as exibe para o usuário. É válido lembrar que a única forma de interagir (configurar e enviar requisições) com o sistema é por meio do aplicativo, não existe um software para PC ou outra forma para isso. Apenas para configuração inicial do sistema é usado um software chamado DS Manager. Este envia pacotes broadcast na rede para encontrar dispositivos que se comunicam dentro de um protocolo específico e então faz as configurações iniciais da placa controladora.

2.2: Especificações técnicas

Os requisitos do projeto foram definidos junto ao pessoal do departamento de desenvolvimento e aos diretores da empresa. Para a definição dos mesmos, foram levadas em conta as funcionalidades já existentes em outros produtos da empresa, a experiência da própria empresa com as principais necessidades dos usuários de sistemas de segurança automatizados e as tecnologias existentes no mercado.

Dentre os requisitos definidos para o sistema, a tabela 1 apresenta quais que o projeto deve atender, assim como suas unidades e valores-meta.

Tabela 1 - Requisitos do projeto

Requisito de Projeto	Unidade	Valor-meta
1 – Versões do aplicativo para os principais sistemas operacionais para smartphones (Android, iOS e Windows Phone).	S/N	Sim
2 – Comunicação sem fio entre aplicativo e sistema.	S/N	Sim
3 – Acesso ao sistema via intranet e internet.	S/N	Sim
4 – Tempo máximo de atualização da tela do sistema	s	3
5 – Pontos de controle de acesso	porta	2
6 – Pontos de monitoramento de alarme	sensor	1
7 – Quantidade de usuários do sistema	usuário	1 a 10
8 – Avisar o usuário via SMS	S/N	Sim
9 – Quantidade máxima de celulares cadastrados para avisos do sistema	celular	1 a 10
10 – Botão de pânico	S/N	Sim
11 – Registro de eventos	S/N	Sim
12 – *Reset* físico	S/N	Sim

Android, iOS e Windows Phone são os três sistemas operacionais para telefones móveis com maior participação no mercado. Juntos possuem mais de 70% dos smartphones no mundo rodando um desses sistemas operacionais e essa parcela tende a crescer futuramente [5]. Logo, disponibilizar um produto que seja compatível com esses sistemas operacionais deve atender a maioria dos clientes em potencial, como prevê o requisito 1. No entanto, este trabalho só abrange o desenvolvimento do aplicativo para Android, o qual será melhor detalhado no capítulo 3.

O requisito 2 visa garantir mobilidade ao usuário permitindo que o mesmo possa interagir com o sistema de qualquer ponto, basta estar com o aparelho móvel em mãos. O requisito 3 complementa o 2, pois faz como que o

usuário possa ter acesso ao sistema tanto dentro de sua residência como remotamente em qualquer outro ponto em que tenha acesso à internet.

A atualização da interface gráfica do aplicativo deve acontecer em um período constante para que o usuário saiba como está o estado do sistema. O requisito 4 exige que essa atualização seja feita a cada 3 segundos.

No que diz respeito aos pontos de controle de acesso, o sistema deve suportar 2 pontos para cumprir o requisito 5. Um ponto será uma porta com uma fechadura eletromagnética e outro será um portão eletrônico.

Para monitoramento de ambientes contra invasões o sistema vai monitorar um sensor de presença, como definido no requisito 6.

Para cumprir o requisito 7, o sistema deve dar acesso para vários usuários.

Os avisos de alarme, comumente feitos por sinais sonoros acompanhados ou não de sinais luminosos, serão realizados por meio de mensagens SMS de alerta para celulares cadastrados no sistema. Então, para atender o requisito 8, o sistema deve oferecer suporte para envio de mensagens SMS. O requisito 9 está relacionado ao 8, pois o envio de SMS deverá ser feito para um ou mais números de celulares (máximo 10) presentes em uma lista de cadastro.

O botão pânico, definido no requisito 10, é um botão existente no aplicativo que serve para situações de emergência. Ao acioná-lo, por qualquer usuário, o sistema envia um SMS de emergência para todos os celulares cadastrados na lista de aviso do botão de pânico.

Salvar um histórico dos eventos é uma funcionalidade comum em sistemas de segurança. Pois ter um log de registros permite que esses dados possam ser usados para análises dos usuários ou até de uma eventual investigação. Portanto, como prevê o requisito 11, o sistema deve guardar seus principais eventos para serem consultados pelo aplicativo.

O requisito 12 apenas define a existência de um *reset* no sistema para que, quando acionado, o mesmo volte às configurações de fábrica.

2.3: Módulos funcionais

Uma vez definidos os requisitos do sistema, o passo seguinte foi traduzir o mesmo do ponto de vista modular e funcional. Isso facilita na busca por soluções, já que as características do problema estão melhores definidas.

A função global do sistema, como já mencionado anteriormente, é controlar e monitorar o acesso de pessoas a um determinado ambiente. Em outras palavras, quer-se que o sistema garanta ao usuário total controle de entrada e saída de pessoas em sua residência.

Dessa maneira, podemos modelar o problema como mostra a figura 3 e então dividi-lo em módulos para melhor compreensão.

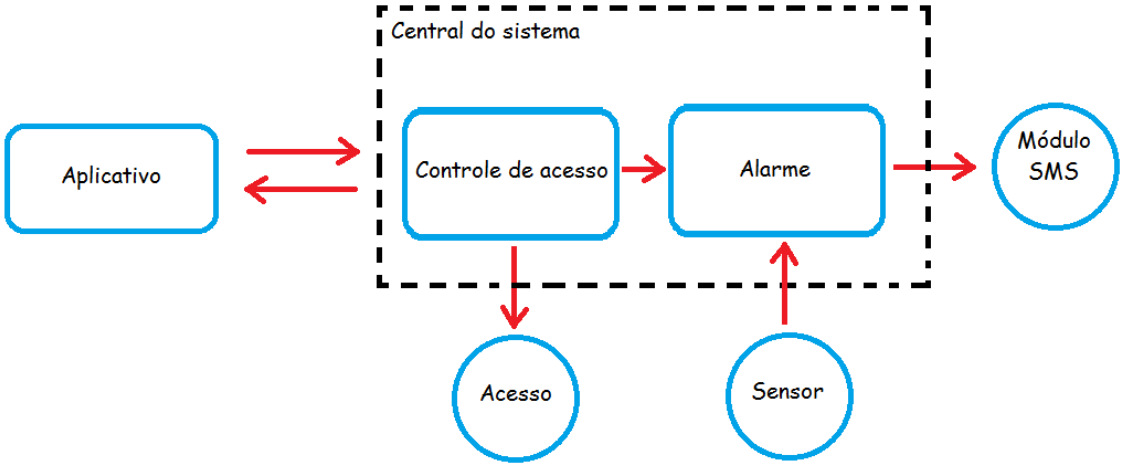

Figura 3 - Esquema dos módulos do sistema

As funções do aplicativo são enviar comandos de requisição de acesso e ativação/desativação do alarme. É também tarefa do aplicativo solicitar informações do sistema e exibi-las para o usuário. O aplicativo deve ter uma interface clara e objetiva, ou seja, exibir para o usuário apenas as informações essenciais do sistema. Os detalhes de desenvolvimento do aplicativo serão apresentados no capítulo 4.

A central do sistema deve prover a monitoração e o controle dos outros módulos, além de atender as requisições do aplicativo. Essa central será

implementada em uma placa controladora (NetControl). Também faz parte do escopo deste trabalho o desenvolvimento do *firmware* dessa placa e será mais bem detalhado no capítulo 5.

Para integrar o aplicativo com a placa controladora para que haja troca de mensagens entre os mesmos usou-se comunicação via TCP/IP. A placa NetCrontrol possui um módulo *ethernet* com suporte a conexões TCP/IP. Já o aplicativo usa o módulo wi-fi, existente em praticamente todos os smartphones, para trocar mensagens na rede. A arquitetura usada para integração entre o aplicativo e a placa foi a cliente-servidor, sendo a placa a servidora e o aplicativo fazendo o papel do cliente. A parte de integração do sistema será apresentada no capítulo 6.

Tendo em vista o que foi definido e exposto, dividiu-se o projeto em duas frentes de trabalho, um responsável por desenvolver o aplicativo para smartphone e outra por desenvolver o *firmware* da central do sistema. Essas duas frentes foram trabalhadas de forma paralela e os detalhes de implementação assim como os resultados serão mostrados nos próximos capítulos.

Capítulo 3: Plataformas móveis

Neste capítulo serão mostradas questões relativas aos sistemas operacionais para smartphone. Aqui serão apresentados os resultados sobre uma pesquisa feita pelo autor deste documento sobre sistemas operacionais para telefones móveis. Após a pesquisa foram tomadas algumas decisões de projeto junto ao gerente de desenvolvimento e aos diretores da empresa a respeito de que plataformas móveis o sistema irá oferecer suporte. Decidiu-se também qual sistema operacional faria parte da primeira versão do aplicativo, o qual faz parte do escopo deste trabalho.

Além disso, serão mostradas as características do sistema operacional escolhido que precisaram ser estudadas para desenvolver o aplicativo.

3.1: Escolha do sistema operacional

A primeira atividade feita antes de começar a desenvolver o aplicativo foi uma pesquisa sobre os principais sistemas operacionais existentes no mercado. A pesquisa teve o objetivo de levantar as características de cada sistema operacional, o custo com desenvolvimento, ferramentas e facilidade de portabilidade para outros sistemas operacionais. Teve-se a necessidade de fazer essa pesquisa para saber como está dividido o mercado das plataformas móveis para smartphones.

Dentre as plataformas para dispositivos móveis existentes, as que dominam praticamente todo o mercado são: iOS, Android, Windows Phone, BlackBerry OS e Symbian [5].

3.1.1: iOS

O iOs é um sistema operacional para dispositivos móveis da Apple. Esse sistema operacional roda apenas em produtos da empresa como iPhone, iPod e iPad. A Apple não permite que o iOS seja rodado em hardware de terceiros.

É baseado no sistema operacional Mac OS X, usado nos computadores Macintoch, que por usa vez é baseado no UNIX BSD.

Para desenvolver aplicativos para iOS deve-se fazer um cadastro no site da Apple. O cadastro e o download do SDK (Software Development Kit) podem ser feitos gratuitamente e pode-se desenvolver e testar os aplicativos em um simulador. Para testar em dispositivos reais e vender os aplicativos na App Store é preciso solicitar uma licença que tem uma taxa anual de US$ 99,00. A licença garante outros benefícios como suporte técnico e acessos a versões beta do iOS e da SDK [7].

Itens necessários para desenvolver:

- Mac para rodar o iOS SDK.
- iOS Developer Program (US$ 99,00 / ano).
- XCode IDE (gratuita).
- Linguagem Objective-C.
- “iDevice” (iPhone, iPod, iPad) registrado.

3.1.2: Android

Android é um sistema operacional que roda sobre o núcleo Linux. Inicialmente foi desenvolvido pela Google e posteriormente pela Open HandSet Alliance. O Android permite que suas aplicações sejam desenvolvidas em linguagem de programação Java controlando os dispositivos usando bibliotecas criadas pela Google [8].

Em outubro de 2011, existiam mais de 500 mil aplicações rodando sobre a plataforma Android [9].

Por ser um sistema operacional baseado em Linux, que é Multi-Thred por origem, o Android é capaz de executar vários aplicativos e processos ao mesmo tempo.

Itens necessários para desenvolver:

- PC rodando Windows, Linux ou até mesmo um Mac.
- IDE com suporte ao Android SDK (Eclipse ou NetBeans).

- Android SDK (download gratuito no site do Android).
- Linguagem java.
- Smartphone rodando Android.

3.1.3: Windows Phone

Windows Phone é um sistema operacional para telefonia móvel da Microsoft e surgiu para substituir o Windows Mobile, apesar de não ser compatível com o mesmo.

Com o poder de flexibilidade do .NET Framework, não foi necessário a Microsoft criar uma nova linguagem de programação para plataforma móvel. Contudo, foi feito uma customização no framework para suportar a nova plataforma, e atualmente pode-se escolher entre as linguagens C# e Visual Basic (VB) para criação de aplicativos.

Itens necessários para desenvolver [10]:

- IDE Visual Studio for Mobile Phone (disponibilizado gratuitamente pela Microsoft).
- SDK (disponibilizada gratuitamente pela Microsoft).
- Linguagem C# ou Visual Basic.
- Smartphone rodando Windows Phone.

3.1.4: BlackBerry OS

Black Berry OS é um sistema operacional móvel desenvolvido pela empresa RIM (Research In Motion) para dispositivos Black Berry. Inicialmente essa plataforma foi desenvolvida para sistemas de mensagens. Mas atualmente ela oferece um ambiente de telefone, edição de texto, e-mail, além de recursos de mídia e GPS presentes em dispositivos mais novos. Toda a comunicação de dispositivos Black Berry com a internet é feita por meio de centro de dados RIM com o uso de um protocolo de segurança.

Itens necessários para desenvolver [11]:

- PC rodando Windows, Linux ou até mesmo um Mac.
- Black Berry JDE.
- Linguagem java.
- Smartphone Black Berry.

3.1.5: Symbian

Symbian OS é um sistema operacional para telefones móveis com suporte para vários recursos como câmera fotográfica, wireless, bluetooth, MMS, entre outras funções.

O Symbian é um consórcio de várias empresas como Nokia, Siemens, Samsung, Ericsson, Sony Ericsson e Panasonic. Empresas que não pertencem ao consórcio podem comprar a licença para utilizar o Sistema Operacional em seus produtos.

É um sistema aberto e de baixo custo, possui recursos para gerenciar e utilizar pouca bateria e memória. Permite a instalação de softwares de terceiros e aproveita bem os recursos do aparelho como memória RAM, processador, processador gráfico, etc.

O Symbian é um sistema operacional bastante versátil, pois permite que seus aplicativos sejam desenvolvidos em várias linguagens como: Symbian C/C++, Java ME, FlashLite, HTML5, entre outras [12].

Itens necessários para desenvolver:

- PC rodando Windows ou Linux.
- IDE com suporte.
- Symbian OS SDK.
- Linguagem C++ ou Java.
- Smartphone rodando Symbian.

3.1.6: Quadro comparativo

De posse das principais características de cada sistema operacional, foi feito um quadro comparativo que pode ser visto na tabela 2.

Tabela 2 - Quadro comparativo das principais plataformas móveis do mercado

	iOS	Android	Windows Phone	Black Berry	Symbian
Custo da licença para programar	99$ / ano	0	0	0	0
IDE e SDK	gratuita	open source	gratuita	gratuita	gratuita
Variedade de opções de hardware	Baixa	Alta	Média	Média	Média
Linguagem de programação	Objective - C	Java	C# ou Visual Basic	Java	Symbian C++ e outras

Observando esse quadro comparativo, nota-se que o único que tem um custo com licença é o iOS, os demais não possuem. As ferramentas de desenvolvimento são gratuitas para todos eles.

A linguagem de programação usada para desenvolver os aplicativos é diferente em quase todos com exceção do Java para Android e BlackBerry OS que, ainda assim, possuem API's (Application Programming Interface) e rodam em máquinas virtuais diferentes. Enquanto os aplicativos Black Berry rodam sobre a máquina virtual do Java ME, os aplicativos Android rodam sobre a máquina virtual Dalvik que é própria para o sistema operacional. Isso faz com que a questão de portabilidade de código de um sistema operacional para outro seja muito difícil.

Existem projetos como o aplicativo BlueStacks, que está em fase de testes no Windows 7, que pretende rodar aplicativos desenvolvidos para Android em Windows Phone. Outro projeto é da Black Berry que pretende

lançar seus próximos celulares com o sistema QNX que promete ser compatível com aplicativos para Android.

A plataforma Android, além de ser *open source* e não ter custos com desenvolvimento, ela tem uma comunidade de desenvolvedores maior. Esses foram fatores que tiveram forte influência na escolha por essa plataforma. Mas, além desses fatores, do ponto de vista comercial o sistema operacional Android é o que possui a maior fatia de mercado dentre os já citados. Logo, desenvolver um aplicativo para essa plataforma significa ter uma gama maior de consumidores do produto, quando ele for lançado no mercado. A seção seguinte mostra alguns dados sobre o mercado de smartphones.

3.1.7: Informações sobre o mercado dos smartphones

Algumas pesquisas mostram que recentemente há um crescimento muito forte no número de vendas de smartphones no mundo. Segundo um estudo publicado pelo instituto de pesquisas Gartner, no último trimestre de 2011 foram vendidos mais de 107 milhões desses aparelhos no mundo, um aumento de 73,6% em relação ao mesmo período do ano anterior. Os smartphones representaram 25% das vendas totais dos aparelhos celulares [5].

Os números dessa pesquisa podem ser observados na tabela 3.

Tabela 3 - Vendas de dispositivos móveis no mundo no segundo trimestre de 2011 (milhares de unidades)

Empresa	**2º Trimestre 2011**	**2º Trimestre de 2011 – Market Share (%)**	**2º Trimestre 2010**	**2º Trimestre de 2010 – Market Share (%)**
Nokia	97869.3	22.8	111473.7	30.3
Samsung	69827.6	16.3	65328.2	17.8
LG	24420.8	5.7	29366.7	8.0
Apple	19628.8	4.6	8743.0	2.4
ZTE	13070.2	3.0	6730.6	1.8
RIM	12652.3	3.0	11628.8	3.2
HTC	11016.1	2.6	5908.8	1.6
Motorola	10221.4	2.4	9109.4	2.5
Huawei Device	9026.1	2.1	5276.4	1.4
Sony Ericsson	7266.5	1.7	11008.5	3.0
Others	153662.1	35.8	103412.6	28.1
Total	**428661.2**	**100.0**	**367986.7**	**100.0**

Em se tratando de sistemas operacionais rodando em telefones móveis, a distribuição em valores absolutos de vendas em 2011 com relação a 2010 pode ser vista na tabela 4, também retirada da mesma pesquisa.

Tabela 4 - Vendas de smartphones no mundo por sistemas operacionais (milhares de unidades)

Sistema operacional	**2º Trimestre 2011**	**2º Trimestre de 2011 – Market Share (%)**	**2º Trimestre 2011**	**2º Trimestre de 2011 – Market Share (%)**
Android	46775.9	43.4	10652.7	17.2
Symbian	23853.2	22.1	25386.8	40.9
iOS	19628.8	18.2	8743.0	14.1
Research In Motion	12652.3	11.7	11628.8	18.7
Bada	2055.8	1.9	577.0	0.9
Microsoft	1723.8	1.6	3058.8	4.9
Others	1050.6	1.0	2010.9	3.2
Total	**107740.4**	**100.0**	**62058.1**	**100.0**

Em participação absoluta, pode-se observar que a plataforma Android tem a maior fatia de participação no mercado, mas ainda não é maioria. A figura 4 mostra como estão distribuídos os sistemas operacionais para dispositivos móveis no mercado.

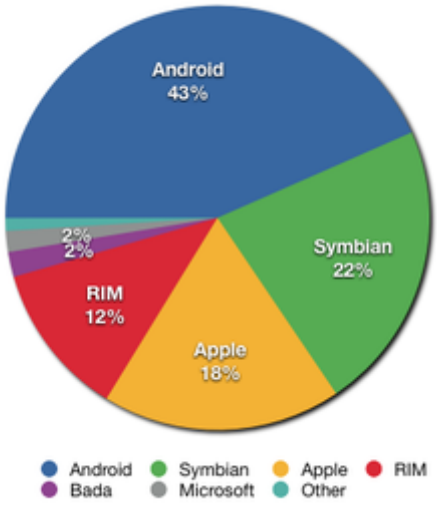

Figura 4 - Distribuição dos sistemas operacionais rodando nos dispositivos móveis

Outro ponto que vale ser lembrado é que a Nokia, em parceria com Microsoft, lançará seus próximos modelos de smartphones com Windows Phone. Apesar desse sistema operacional não aparecer nesse gráfico, sua participação no mercado crescerá nos próximos anos, pois a Nokia até o momento é a empresa que mais vende celulares no mundo.

Segundo a Gartner, a tendência é que em 2015 o sistema operacional Android esteja instalado em metade dos smartphones vendidos. Os dados dessa estimativa podem ser vistos na tabela 5.

Tabela 5 - Projeção de vendas de dispositivos móveis de comunicação (milhares de unidades)

OS	**2010**	**2011**	**2012**	**2015**
Symbian	111,577	89,930	32,666	661
Market Share (%)	37.6	19.2	5.2	0.1
Android	67,225	179,873	310,088	539,318
Market Share (%)	22.7	38.5	49.2	48.8
Research In Motion	47,452	62,600	79,335	122,864
Market Share (%)	16.0	13.4	12.6	11.1
iOS	46,598	90,560	118,848	189,924
Market Share (%)	15.7	19.4	18.9	17.2
Microsoft	12,378	26,346	68,156	215,998
Market Share (%)	4.2	5.6	10.8	19.5
Other Operating Systems	11,417.4	18,392.3	21,383.7	36,133.9
Market Share (%)	3.8	3.9	3.4	3.3
Total Market	**296,647**	**467,701**	**630,476**	**1,104,898**

Após a pesquisa sobre os sistemas operacionais foi decidido que o produto final deverá ter versões do aplicativo para as plataformas Android, iOS e Windows Phone. A primeira versão, que está dentro do escopo deste trabalho, a ser desenvolvida será para plataforma Android.

3.2: O sistema operacional Android

O Android é um sistema operacional para dispositivos móveis que hoje é desenvolvida e mantida pela Open Handset Alliance, apesar de ainda ser gerenciada pela Google**.** Essa plataforma permite que se criem softwares em linguagem Java usando bibliotecas desenvolvidas pela Google. Desde 2009 a plataforma é open source e a empresa tem feito esforço para manter um programa de compatibilidade (Android Compatibility Program) que define um dispositivo compatível Android, evitando incompatibilidade de implementação de aplicativos.

3.2.1: Características

O sistema operacional Android é definido como uma pilha de softwares. Cada camada da pilha agrupa programas que dão suporte a funções específicas do sistema operacional. O esquema da figura abaixo mostra como é formada essa pilha de software.

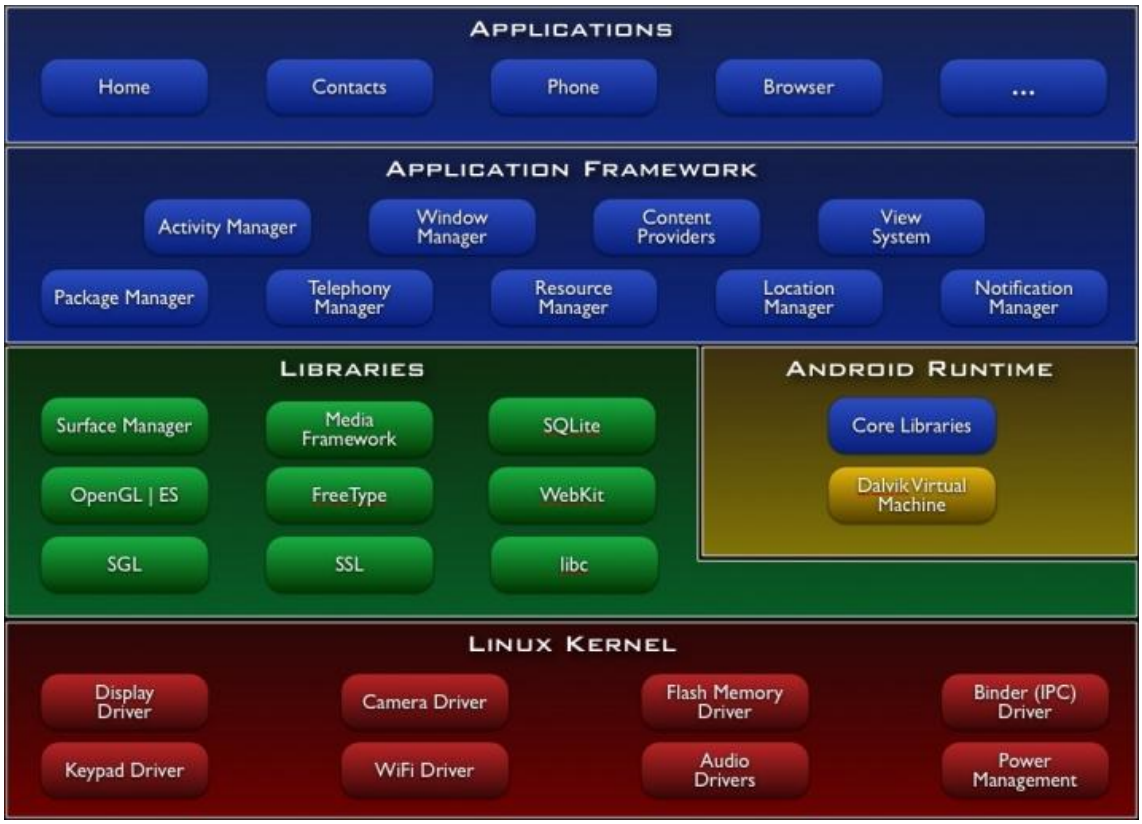

Figura 5- Arquitetura do sistema operacional Android

A camada que fica na base da pilha é o kernel. O kernel do Android foi construído baseado na versão 2.6 do Linux, o qual inclui programas de gerenciamento de memória, configurações de segurança, software de gerenciamento de energia e drivers de hardware. Os drivers são programas que controlam os dispositivos de hardware de cada aparelho como: câmera, acelerômetro, GPS, etc.

A camada Libraries, como o próprio nome diz, é onde estão as bibliotecas básicas do sistema. A maior parte dessas bibliotecas foi desenvolvida em C/C++. Essas bibliotecas têm as funções que fazem os dispositivos trabalharem com vários tipos de dados. Por exemplo, a biblioteca Media Framework oferece suporte para reprodução e gravação de vários formatos de áudio e vídeo, a SQLite possibilita persistência de informações com suporte a banco de dados e a OpenGL trabalha com gráficos.

No mesmo nível da camada Libraries está a Android Runtime. Essa camada inclui a Core Libraries em que se encontram as bibliotecas do núcleo

Java. Também nessa camada se encontra a Dalvik Virtual Machine, onde são executadas as aplicações. Essa máquina virtual é otimizada para utilizar pouca memória e projetada para que múltiplas instâncias suas sejam rodadas ao mesmo tempo, deixando para o sistema operacional o isolamento de processos, o gerenciamento de memória e suporte a threading. Cada aplicação Android é executada em um processo separado, sobre sua própria instancia da máquina virtual.

A Application Framework é uma camada desenvolvida quase toda em Java e provê um conjunto de bibliotecas para acessar os recursos do dispositivo como interface gráfica, localizador (GPS), banco de dados persistente, armazenamento SD, etc. Esta camada que os desenvolvedores usam para construir suas aplicações.

A camada do topo, a Applications, é onde estão as aplicações em si. Nesta camada estão os aplicativos como navegador Web, gerenciador de e-mails e games. As aplicações são feitas em Java e executadas nas máquinas virtuais. É esta camada que é visível ao usuário comum.

3.2.2: Componentes de aplicação

Os componentes da aplicação são blocos de construção essenciais de uma aplicação Android, ou seja, todo aplicativo é composto por esses blocos. Cada componente é um ponto diferente através da qual o sistema pode entrar em sua aplicação. Nem todos os componentes são pontos de entrada reais para o usuário e alguns dependem uns dos outros, mas cada um existe como uma entidade própria e desempenha um papel específico, cada um é um bloco de construção singular que ajuda a definir o comportamento geral do aplicativo [13].

Há quatro tipos diferentes de componentes de aplicação. Cada tipo tem uma finalidade distinta e tem um ciclo de vida diferente que define como o componente é criado e destruído.

Aqui estão os quatro tipos de componentes de aplicação:

- **Activities:** uma Activity representa uma única tela de interface com o usuário. Por exemplo, em um aplicativo de captura de imagens, existe uma Activity para exibição da imagem a ser tirada, outra para gravação de vídeo e outra para visualização dos arquivos. Apesar de elas terem que trabalhar juntas para o funcionamento do aplicativo, uma não depende da outra. Dessa forma, outro aplicativo pode iniciar a Activity de outro (se tiver permissão). Por exemplo, um aplicativo de uma rede social pode chamar uma Activity do aplicativo da câmera para o usuário capturar uma foto que deseja ser compartilhada.
- **Services:** um Service é um componente que roda em background no aplicativo e não possui uma interface gráfica para o usuário. Por exemplo, um Service pode reproduzir uma música enquanto o usuário está usando outro aplicativo ou baixar dados da rede não bloqueando a interação do usuário com o aplicativo. Assim como uma Activity, pode ser iniciado por outro componente.
- **Content Providers:** um Content Provider é um componente que gerencia dados compartilhados do aplicativo. É possível armazenar dados em sistemas de arquivos, banco de dados SQLite, na Web ou em qualquer outro local que o aplicativo possa acessar. Por meio de um Content Provider é possível que o conteúdo de um aplicativo seja acessado e até modificado por outros (se tiverem permissão). Os Content Providers também são úteis para leitura e escrita de dados privados para uma aplicação.
- **Broadcast Receivers:** um Broadcast Receiver é um componente que responde a avisos broadcast do sistema. O sistema pode avisar que a bateria do dispositivo está fraca ou que uma imagem foi capturada e avisar para que as outras aplicações fiquem sabendo. Mas aplicações também podem enviar avisos broadcast. Por exemplo, avisar o término de um download de arquivo para outra aplicação que fará uso do mesmo. Os Broadcasts Receivers não possuem interface com o usuário e servem apenas como um receptor do aplicativo para que outros componentes possam executar seus processos.

Um aspecto único do projeto do sistema Android é que qualquer aplicação pode iniciar um componente de outro aplicativo. Por exemplo, se você deseja que o usuário possa capturar uma foto com a câmera do dispositivo, provavelmente há outro aplicativo que faz isso e sua aplicação pode usá-lo, em vez de desenvolver uma Activity própria para capturar uma foto. Não é preciso incorporar o código do aplicativo da câmera. Em vez disso, simplesmente é iniciada a Activity no aplicativo da câmera que captura uma foto. Quando capturada a imagem, a foto retorna para a sua aplicação para que você possa usá-la. Para o usuário, é como se a câmera realmente fizesse parte de sua aplicação.

Pelo fato de cada aplicação executar em um processo separado com permissões de arquivos que restringem o acesso a outros aplicativos, um aplicativo não pode diretamente ativar um componente de outro aplicativo. O sistema Android, no entanto, pode. Então, para ativar um componente em outro aplicativo, você deve enviar uma mensagem ao sistema que especifica a sua intenção de iniciar um componente particular. O sistema, então, ativa o componente para o seu aplicativo [13].

Capítulo 4: O aplicativo

Este capítulo mostrará como foi feito o desenvolvimento do aplicativo para smartphone do sistema. Aqui será apresentada a metodologia empregada, quais ferramentas foram usadas, o modelo do software do aplicativo, o projeto das interfaces, detalhes de implementação e testes de desenvolvimento.

4.1: Metodologia

A IDE (Integrated Development Environment) usada para programação foi o Eclipse com a SDK (Software Development Kit) do Android, disponível para download no site do sistema operacional e com uma documentação completa que serviu de fonte para este trabalho.

Para desenvolver o software do aplicativo usou-se a seguinte metodologia:

- Primeiramente fez-se um estudo da API Android, dos componentes existentes citados anteriormente e das ferramentas de desenvolvimento.
- Em seguida foram levantadas as características e funções desejadas no aplicativo e quais as informações seriam exibidas para o usuário.
- Foi feito um modelo da estrutura do software para deixá-lo modular para facilitar a implementação, manutenção e garantir flexibilidade para melhorias futuras.
- Depois foi feito o projeto da camada de interface com o usuário.
- E por fim a codificação e testes.

4.2: Ferramentas de gerenciamento de projeto

Para auxílio durante o desenvolvimento e o gerenciamento do aplicativo, foram usadas algumas ferramentas que a Automatiza usa em seus projetos para planejamento e documentação. São elas:

- SVN Subversion: é um repositório de arquivos usado para controle de versão. Ele foi usado para gerenciar o código-fonte, manter histórico de alterações e também manter uma cópia do código-fonte no servidor da empresa, onde é mais seguro.
- Wiki: foram usadas páginas da Wiki da empresa para documentação do projeto e registro de tutoriais de uso de ferramentas novas que foram estudadas para desenvolver este trabalho e contribuir para a base de conhecimento da empresa.
- Mantis Bug Tracker: ferramenta com interface Web para gerenciar melhorias e defeitos de software. Através dessa ferramenta também é feito o planejamento de atividades do projeto dentro de um intervalo de tempo.

4.3: Diagrama de classes

Após o estudo sobre as ferramentas de desenvolvimento e de posse das características que o aplicativo devia ter, foi feito um modelo UML do software, seu diagrama de classes simplificado pode ser visto na figura 6.

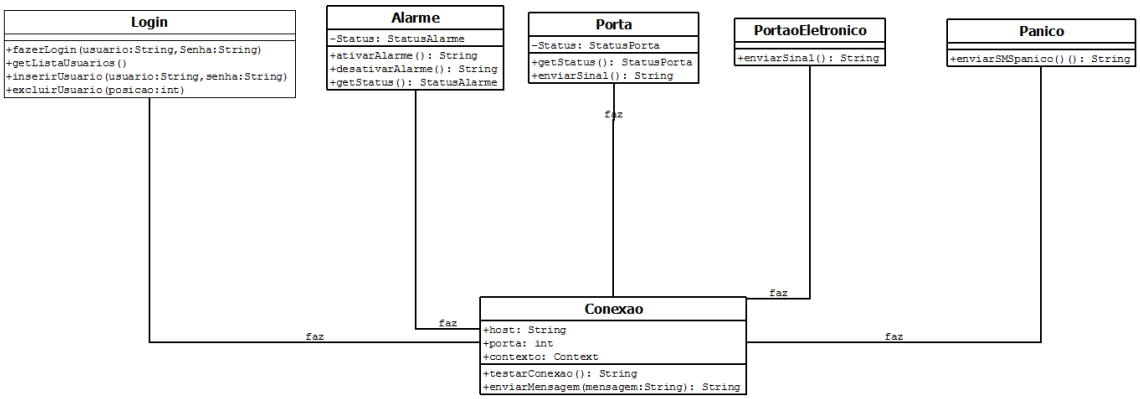

Figura 6 - Diagrama de classes do aplicativo

As cinco classes da parte de cima do diagrama (Login, Alarme, Porta, PortaoEletronico e Panico) instanciam um objeto do tipo “Conexao” para poderem trocar mensagens com a placa. Elas são instanciadas nas classes de interface do aplicativo. Para explicar melhor, a seguir segue uma melhor descrição de cada uma dessas classes.

4.3.1: Classe Conexao

Primeiramente foi escrita uma classe que fizesse a conexão com a placa, a qual é representada pela classe Conexao. Ela possui três atributos que são passados no método construtor da classe:

- Host: esse atributo faz referência ao endereço que o aplicativo vai tentar estabelecer conexão. É esperado um número de IP (XXX.XXX.XXX.XXX).
- Porta: recebe o número da porta de comunicação em que será feita a conexão. Espera um valor inteiro até 65535.
- Contexto: guarda o contexto da aplicação em que o objeto da classe foi instanciado.

Essa classe possui dois métodos:

- testarConexao(): é um método simples que serve só para testar se a conexão está sendo estabelecida e não há problemas na rede.
- enviarMensagem(String mensagem): esse método é o responsável por enviar as mensagens do aplicativo para a placa controladora. Sua função é abrir uma conexão com a placa, enviar a mensagem recebida como parâmetro e retornar a resposta da placa.

As mensagens enviadas seguem um protocolo criado em nível de aplicação. Na descrição dos métodos das próximas classes serão colocadas as mensagens que cada um envia. Mas o protocolo será explicado no capítulo 6 que trata da integração do aplicativo com a placa.

4.3.2: Classe Login

Toda a parte de controle de acesso ao aplicativo é feita por essa classe. Ela é responsável por conferir se as informações de login são válidas e liberar o acesso ao sistema. Esta classe também faz o cadastramento de usuários. Seus métodos são:

- fazerLogin(String nome, String senha): envia uma mensagem para que o sistema confira o login do usuário (mensagem = “ll*nome#senha”).
- getListaUsuarios(): envia uma mensagem pedindo a lista de usuários cadastrados no sistema (mensagem = “lh*”).
- inserirUsuario(String nome, String senha): envia mensagem de cadastro de novo usuário (mensagem = “li*nome#senha”).
- excluirUsuario(int id): envia uma mensagem para deletar o usuário especificado pelo id (mensagem = “ld*id”).

4.3.3: Classe Alarme

Esta classe é responsável por interagir com o módulo de alarme do sistema. Ela instancia um objeto da classe Conexao para enviar as mensagens. Ela possui um atributo:

- StatusAlarme: representa o estado atual do alarme. Pode assumir 3 valores: ativado, desativado ou disparado.

Ela tem 3 métodos:

- ativarAlarme(): envia a mensagem para a ativação do alarme (mensagem = “aa”).
- desativarAlarme(): envia a mensagem para desativar o alarme (mensagem = “ad”).
- getStatus() envia mensagem pedindo o estado atual do alarme (mensagem = “ag”).

4.3.4: Classe Porta

A classe Porta tem a função de enviar a solicitação para a abertura da porta e obter o estado da porta.

Possui um atributo:

- StatusPorta: guarda o estado da fechadura da porta, pode assumir dois estados: aberta ou fechada.

Possui 2 métodos:

- getStatus(): envia a mensagem para obter o estado da porta (mensagem = “ps”).
- enviarSinal(): envia mensagem para que a fechadura seja destrancada. Após um tempo pré-configurado a fechadura tranca novamente, mas isso é tarefa da placa controladora (mensagem = “pp0”).

4.3.5: Classe PortaoEletronico

Esta classe trata da abertura e fechamento do portão eletrônico. Ela possui apenas um método que quando chamado envia uma mensagem para que a placa acione o portão. Ela não possui atributos uma vez que não se pode saber o estado do portão (aberto ou fechado), pois não existe nenhum sensoriamento para isso.

- enviarSinal(): envia mensagem para que a placa acione o portão (mensagem = “ep”).

4.3.6: Classe Panico

É responsável por enviar a requisição para que a placa envie um SMS de aviso de emergência para os celulares cadastrados no sistema. Não possui atributos, apenas um método.

- enviarSMSpanico(): envia requisição de envio de SMS de pânico para os celulares cadastrados (mensagem = “be”).

4.4: Interface

O projeto da camada de interface foi feito usando a metodologia apresentada em [14]. Essa metodologia divide a camada de interface em duas subcamadas: apresentação e aplicação.

4.4.1: Camada de apresentação

Esta camada contém as classes que representam os objetos gráficos da interface e são basicamente responsáveis por receber dados e comandos do usuário e apresentar os resultados a ele.

No desenvolvimento de aplicativos para Android, as interfaces gráficas são desenvolvidas em arquivos .XML (e**X**tensible **M**arkup **L**anguage). Esses arquivos são referenciados dentro do projeto por uma classe chamada R, essa classe é gerada automaticamente com as referências de cada componente gráfico criado nos arquivos .XML. Após a criação de cada interface gráfica, as mesmas são carregadas dentro das Activities que, como citado anteriormente, são os componentes responsáveis por manipular e apresentar as interfaces para o usuário.

O aplicativo do sistema possui basicamente 4 interfaces. Após a instalação do mesmo, a primeira interface que aparece para o usuário é a das configurações iniciais para estabelecer conexão com a placa. A segunda interface é de Login para que o usuário entre com suas informações para solicitar o acesso ao sistema. A terceira é a interface principal, que é onde o usuário controla os dispositivos do sistema: porta, portão eletrônico, alarme e botão de pânico. A quarta é a interface de configurações do sistema. Esta possui 4 abas, uma para cada dispositivo citado anteriormente. A figura abaixo mostra essas quatro interfaces gráficas.

Figura 7 - Interfaces do aplicativo

4.4.2: Camada de aplicação

A camada de aplicação controla a lógica da interface, ou seja, quais as sequências de carregamento das interfaces e quais os eventos determinam que o usuário navegue de uma interface para outra.

O diagrama de estados de navegação do aplicativo pode ser visto na figura 8.

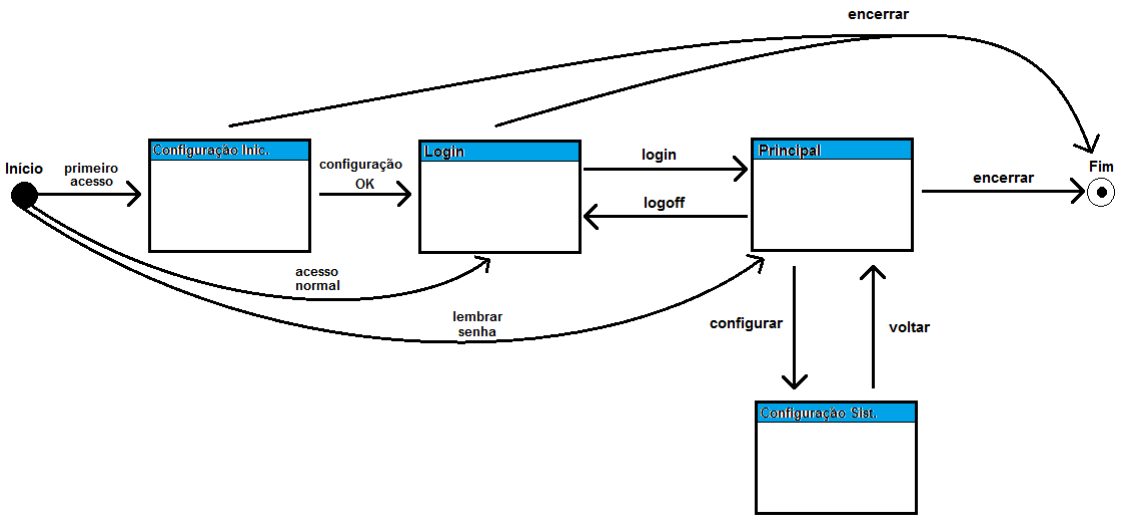

Figura 8 - Diagrama de estados de navegação

Cada interface é manipulada por uma classe herdeira de uma Activity. Logo a manipulação dos componentes gráficos e a lógica de mudança das telas são tratadas por essas classes. Essas não foram colocadas no diagrama de classes porque estão associadas à camada de interface.

Em um primeiro acesso, logo após a instalação, a primeira tela que aparece para o usuário é a de configurações iniciais. Nesta tela que o usuário, além de nome e senha, informa o endereço IP da placa e a porta de conexão para que o aplicativo possa estabelecer conexão. Realizado um primeiro acesso, essa tela deixa de aparecer nas próximas vezes que o aplicativo for aberto. A Activity responsável por esta interface instancia um objeto do tipo Login para manipular as informações fornecidas pelo usuário e conectar com a placa.

A sequência principal das interfaces começa com a tela de Login. À partir daí pode-se encerrar o aplicativo ou, para seguir para a tela principal, o usuário deve informa um login e senha válidos. Um objeto do tipo Login, instanciado pela Activity responsável por esta interface, também realiza o tratamento das informações de acesso ao aplicativo.

Na tela Principal pode-se fazer logoff e voltar para a tela de Login caso queira trocar de usuário, pode-se abrir a tela de configuração ou encerrar o aplicativo. A Activity desta interface instancia quatro objetos, um de cada uma das quatro classes responsáveis pelos dispositivos do sistema.

Na tela de configuração pode-se navegar pelas 4 abas de configuração do sistema e voltar para a tela Principal. Cada aba desta interface, que também são representadas cada uma por uma Activity, instancia um objeto do dispositivo a ser configurado.

O início do programa pode ser direto na tela Principal. Isso só ocorrerá caso o usuário marque a opção de lembrar senha em um login feito anteriormente. Apesar do aplicativo iniciar na tela Principal, o processo de Login também é realizado neste modo, porém sem que a interface seja exibida.

4.5: Codificação e testes

O projeto de um aplicativo Android tem seus arquivos estruturados da seguinte maneira:

- Diretório src: neste diretório é que são criadas as classes Java da aplicação, todas as classes citadas anteriormente foram criadas aqui.
- Diretório gen: este diretório não pode ser editado, é criado automaticamente pelo compilador. Nele é gerada uma classe “R.java” que possui as referências para os componentes gráficos do aplicativo.
- Diretório res: aqui são criados os arquivos “xml” das interfaces gráficas do aplicativo. Neste diretório também são colocadas as imagens exibidas nas interfaces.
- Arquivo “AndroidManifest”: este arquivo carrega as configurações principais e das permissões do aplicativo.

Para testes durante o desenvolvimento, a própria SDK do Android oferece a possibilidade de emular um smartphone no computador para instalar e executar os aplicativos nele. Além de um aparelho virtual o aplicativo também pode ser instalado em um smartphones real e executado no mesmo por meio de um cabo USB. Pode-se debugar o software do aplicativo rodando direto no smartphone também. Dois modelos foram usados para esse fim: um Galaxy GT-S5360 e um Galaxy GT-I9100, ambos da Samsung.

Capítulo 5: Firmware

Neste capítulo serão abordados os detalhes de desenvolvimento do firmware do sistema. Serão mostradas as ferramentas usadas para desenvolver, as características do hardware e do sistema operacional embarcado onde roda a aplicação e, por fim, a aplicação em si.

Toda a aplicação do firmware foi feita pelo autor deste trabalho que fez uso de várias bibliotecas e rotinas desenvolvidas pela empresa. O pessoal de desenvolvimento de hardware auxiliou no aprendizado do funcionamento do da placa controladora, no uso das ferramentas de desenvolvimento e das rotinas do FreeRTOS e da placa.

5.1: Hardware

O firmware desenvolvido roda sobre uma plataforma de hardware da Automatiza, a placa controladora NetControl. Os motivos de escolha desse hardware foram por ele ser um produto da própria empresa e por ter quase todos os recursos necessários para atender aos requisitos do projeto. No entanto, futuramente, a empresa pretende desenvolver um hardware dedicado a esse sistema e que tenha um módulo GPRS integrado para envio de SMS, pois nesse primeiro protótipo está sendo usado um modem para esse fim.

Entre os recursos que a placa NetControl possui pode-se citar:

- Microcontrolador LC2368 com arquitetura ARM7
- 4 canais de comunicação serial RS232
- 16 canais I/O digitais
- Interface de rede Ethernet
- Memória EEPROM (possui dois CI´s, um 24LC64 com 8 Kb e um 24AA02E48 com 128 bytes de memória)
- Cartão de memória SD
- 2 relés

A figura 9 mostra uma placa NetControl que foi usada no projeto.

Figura 9 - Placa NetControl

5.2: FreeRTOS

As rotinas desenvolvidas para o firmware rodam sobre o FreeRTOS. O FreeRTOS é um sistema operacional de tempo real para sistemas embarcados. Ele tem suporte para 31 microcontroladores, inclusive o LC2368, e é um projeto Open Source distribuído sobre uma licença GPL (General Public License) modificada que permite seu uso para aplicações comerciais [15].

O kernel do FreeRTOS foi projetado para pequenos sistemas embarcados. Suas principais características são:

- Opção de configuração de multitarefa cooperativa, preempitiva e híbrida.

- Leve e fácil de usar. Todo o sistema operacional ocupa cerca de 4 Kb a 9 Kb de memória.
- Estrutura de código que facilita a portabilidade. Quase todo escrito em linguagem C.
- Suporte a multitarefas e recursos de sincronização como *mutexes* e semáforos.
- Nenhuma restrição de software para quantidade de tarefas criadas.
- Eficiente suporte a *timers*.
- Opção para detecção de estouro de pilha.

Como o FreeRTOS já era usado em outros produtos da empresa, foi simples de conhecer seu uso e as ferramentas de desenvolvimento das aplicações. Além disso, também é oferecida uma documentação muito útil no site do FreeRTOS [15].

5.3: Bibliotecas

A comunicação da placa pela rede é feita sobre o protocolo TCP/IP. Para isso, foi usada uma biblioteca open source para empilhamento TCP/IP em sistemas embarcados chamada uIP. A implementação da uIP foi projetada somente para ter os recursos mínimos necessários para uma pilha completa TCP/IP. Ela só dá suporte a uma única interface de rede e implementa os protocolos IP, ICMP, UDP e TCP. A uIP é escrita em linguagem de programação C [16].

Também foram usadas outras bibliotecas da própria empresa para manipulação dos periféricos da placa como: canais de comunicação serial, memória EEPROM, relés e I/O digitais. Essas bibliotecas também foram escritas em linguagem de programação C e foram criadas em projetos anteriores da Automatiza.

5.4: Ferramentas

Para o desenvolvimento do código do firmware foi utilizado um conjunto de ferramentas as quais representam o ambiente de desenvolvimento:

- Eclipse Galileu: é uma IDE open source que, com o uso de plug-ins, pode-se desenvolver software em várias linguagens como Java, C/C++, PHP, Phiton, etc. A compilação e geração do código binário eram feitos por um compilador GCC (GNU Compiler Collection) para linguagem de programação C.
- Docklight: é um software gratuito para teste, análise e simulação de protocolos seriais de comunicação via COM (RS232, RS422/RS485), TCP (cliente e servidor) e UDP. Essa ferramenta foi muito usada para debug, usando uma porta serial da placa para imprimir os estados das variáveis dos processos, facilitando a depuração de erros.
- Flash Magic: é uma ferramenta padrão para gravação de microcontroladores. Após a compilação do código do firmware, carregava-se o arquivo binário nesse software para ser gravado na placa.

5.5: Implementação do código

A estrutura de arquivos do projeto está dividida da seguinte forma. O diretório principal “Source” está divido em quatro diretórios juntos com os arquivos do firmware:

- External: onde estavam os pacote do sistema operacional FreeRTOS e as funções do uIP, arquivos importados para o projeto.
- TCPIP: neste diretório estão os arquivos que configuram os processos de comunicação tanto TCP como UDP.
- PORT: neste diretório estão as funções de acesso aos recursos da placa como configurações do processador LC2368, memória EEPROM, SD Card, canal de comunicação serial, etc.

- UTIL: contém alguns arquivos com funções úteis usadas em outros projetos como leitura/escrita de volumes maiores de dados na EEPROM, estruturas de tempo mais complexas como data, etc.
- Arquivos da aplicação: os arquivos “.c” e “.h” da aplicação do firmware.

Todas as funções do firmware são realizadas em cinco processos que são executados de forma concorrente. Um processo é responsável por atender as requisições do aplicativo e outros quatro são responsáveis por monitorar cada um dos dispositivos do sistema (porta, portão eletrônico, alarme e botão de pânico). A figura 10 mostra um diagrama de representação geral do funcionamento dos processos.

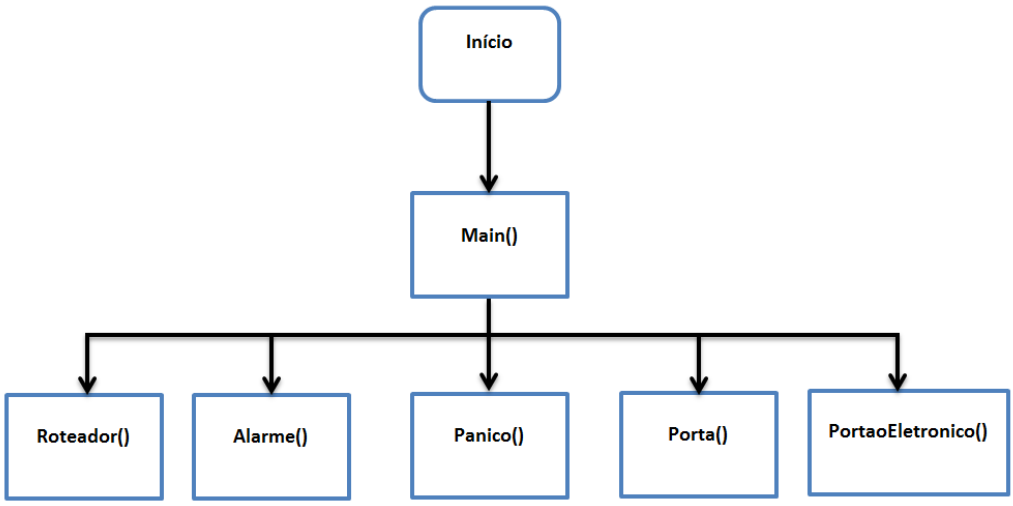

Figura 10 - Diagrama de funcionamento dos processos do firmware

Ao iniciar a execução do firmware, o processo principal Main() executa algumas configurações de hardware e inicializa algumas variáveis do sistema como IP, máscara de rede, gateway e porta de comunicação. Em seguida ele passa a ouvir a porta de comunicação previamente configurada. Após isso, ele instancia os 4 processos dos dispositivos e passa a exercer o processo Roteador().

5.5.1: Processo Roteador()

O processo Roteador() é orientado a eventos. Esse processo fica bloqueado ouvindo a porta de comunicação pré-configurada a espera de uma mensagem vinda do aplicativo. Ao receber uma mensagem, a função *uip_appcall()* é invocada. Essa função é definida na biblioteca uIP citada anteriormente. Dentro da função *uip_appcall()* é chamada então a função *roteadorMensagem_uipappcall (char *mensagem)* que processa a mensagem, identifica para que dispositivo do sistema é a requisição e envia a mesma para ser tratada pelas funções correspondentes àquele dispositivo. A identificação da requisição da mensagem é feita de acordo com o protocolo de mensagens que será explicado no próximo capítulo. Para esclarecer melhor o funcionamento veja figura 11.

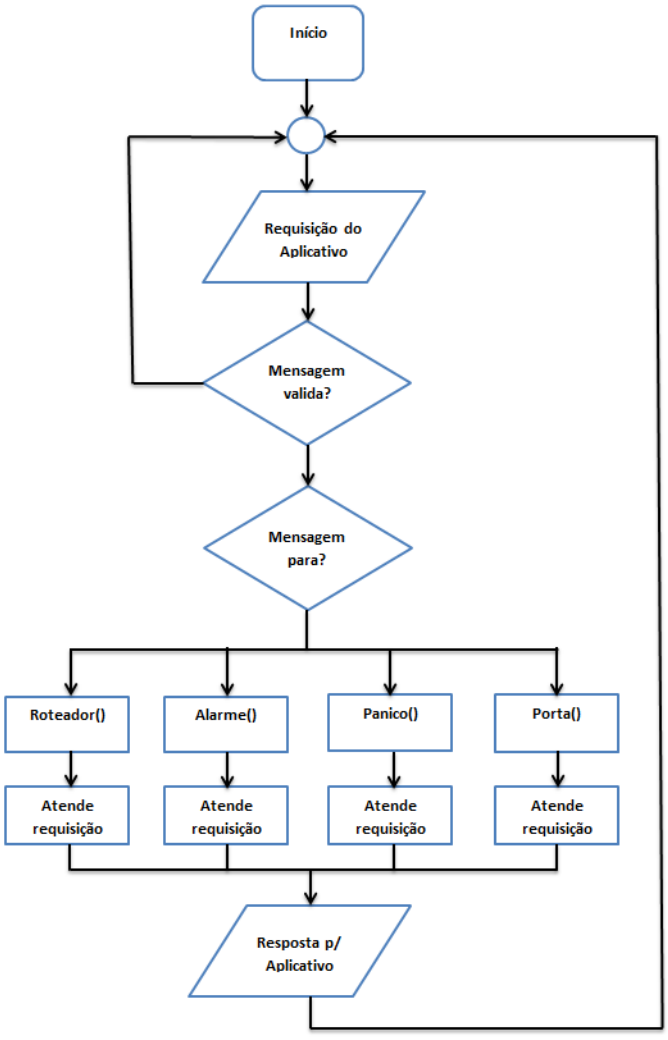

Figura 11 - Fluxograma do processo de roteamento de mensagem

Uma vez que o processo recebe uma mensagem, é feita a identificação da mesma. Confere-se a validade da mensagem, identifica-se o dispositivo, que tipo de requisição foi feita e quais argumentos foram passados.

Para facilitar a organização das requisições, para cada dispositivo existe um arquivo com as funções que atendem suas requisições. As funções da porta estão definidas em manipuladorPorta.c, as funções do portão eletrônico em manipuladorPortaoEletronico.c, as do alarme em manipuladorAlarme.c e do botão de pânico em manipuladorBotaoPanico.c.

As rotinas definidas nesses arquivos que alteram as variáveis de estado do sistema como estado do alarme, abertura da porta e portão eletrônico e envio de mensagem de pânico. Elas também são responsáveis por outras funções como cadastramento de usuários e celulares, configurações internas como tempo de abertura da fechadura e mensagens SMS de aviso. Os dados que devem persistir no sistema são gravados na memória EEPROM da placa.

5.5.2: Processo Alarme()

Este processo é responsável pela monitoração do alarme sistema. Seu ciclo de execução é um loop infinito que fica checando o estado do alarme que pode ser ativado, desativado e disparado. Esse ciclo está representado na figura 12.

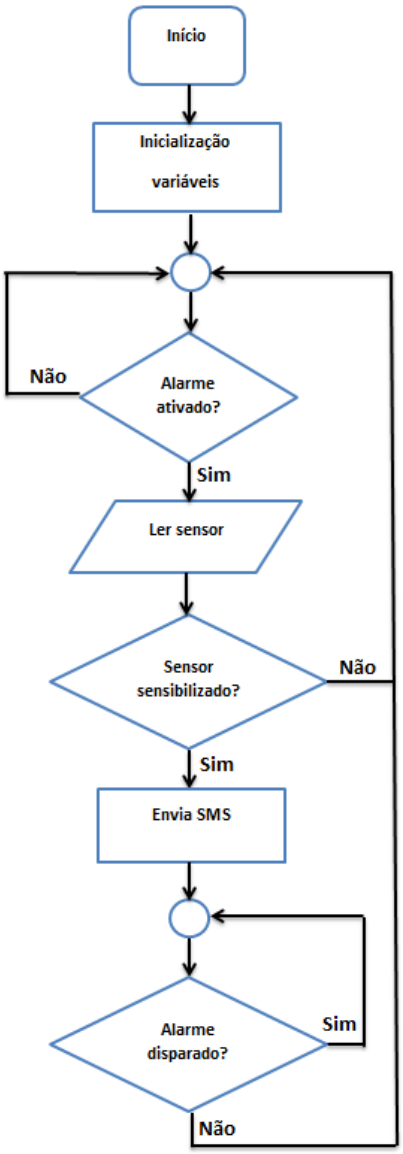

Figura 12 - Fluxograma do processo do alarme

Primeiramente é feita a inicialização das variáveis, logo em seguida o processo entra em loop. A primeira tomada de decisão diz respeito a ativação do alarme, que é representada por uma variável interna (*statusAlarme)* que é carregada da memória EEPROM na inicialização das variáveis. Caso a variável seja 0 (alarme desativado), o processo volta e checa novamente se o alarme foi ativado. Caso o alarme seja ativado a variável *statusAlarme* passa a ser 1 e o processo começa a ler a entrada do sensor de presença ligado à placa. Esse sensor é ligado a uma entrada I/O (ALM) digital da NetControl. Na segunda tomada de decisão, caso o sensor esteja sensibilizado, é chamada uma rotina

que envia SMS para os celulares previamente cadastrados no sistema avisando que o alarme foi disparado. O valor de statusAlarme passa a ser 2 (disparado) e o processo fica preso no último loop até que o alarme seja desativado pelo usuário.

5.5.3: Processo Panico()

Este é o processo responsável pelo botão de pânico. O seu ciclo é simples, apenas confere se o botão foi acionado. Inicialmente são carregadas algumas variáveis como a lista de celulares para os quais vão ser enviadas as mensagens SMS. Uma variável chamada *enviaSMSpanico* representa a ordem de envio do SMS. Caso a variável enviaSMSpanico seja igual a 0, nada é feito e o ciclo volta ao início. Caso seja igual a 1, são enviadas as mensagens SMS a todos os celulares previamente cadastrados. O fluxograma mostrado na figura 13 representa o ciclo de execução deste processo.

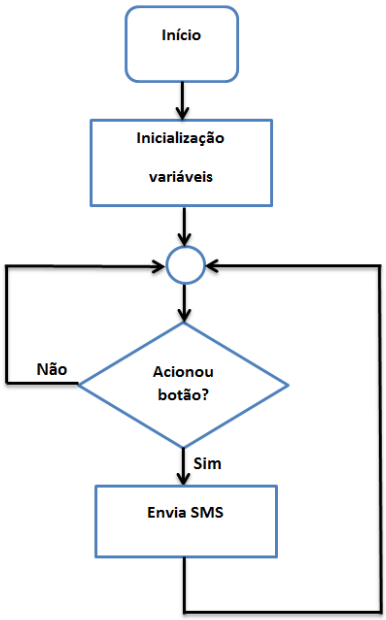

Figura 13 - Fluxograma do processo do botão de pânico

5.5.4: Processo Porta()

Este processo controla a abertura e fechamento de uma fechadura, que pode ser eletromagnética ou elétrica. Ambas podem ser controladas pelo acionamento de um relé. No caso deste sistema, um relé da placa é dedicado ao controle da fechadura.

O ciclo de execução deste processo se resume em conferir se houve alguma requisição para abertura da porta. Essa conferência é feita por meio da variável *sinalPulsoFechadura*. Caso esteja com valor 0 a porta deve permanecer trancada, caso contrário deve-se abrir a fechadura. O ciclo de execução deste processo é mostrado na figura 14.

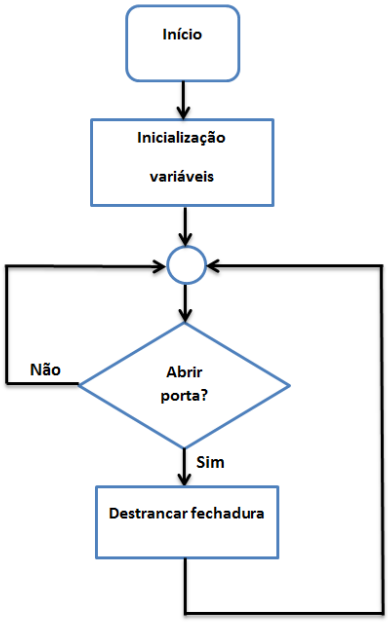

Figura 14 - Fluxograma do processo da porta

Por questão de segurança, o estado normal da fechadura deve ser trancado. Logo, assim que recebe uma requisição para destrancar a porta, a chave do relé abre por um tempo (previamente configurável de 1 a 99 s) para que a fechadura destranque e, em seguida, fecha novamente.

5.5.5: Processo PortaoEletonico()

Este processo tem função muito semelhante ao descrito anteriormente, salvo a exceção de que o tempo de abertura do relé é menor (300 ms), caracterizando apenas um pulso elétrico.

A maioria dos portões eletrônicos no mercado é acionada, além de um controle que emite sinal de radiofrequência, por um contato seco existente no hardware de acionamento do equipamento. Logo foi preciso fazer com que este processo apenas gerasse um pulso no outro relé da placa quando recebesse uma solicitação de abertura de portão eletrônico. Essa técnica já era usada em outros produtos da empresa para acionamento de portões eletrônicos.

O ciclo de execução é mostrado na figura 15.

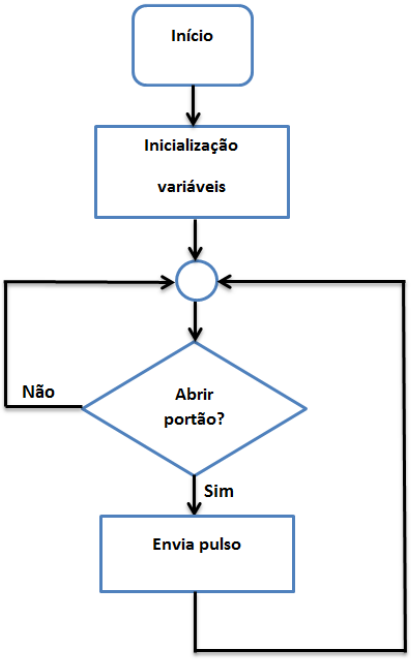

Figura 15 - Fluxograma do processo do portão eletrônico

5.6: Módulo GSM

Como já visto anteriormente, o sistema contém uma função que é o envio de mensagens SMS de aviso, tanto para alarme como para pânico. Como a placa NetControl não dispõe de recursos para resolver esse problema, outras alternativas tiveram que ser exploradas. Dentre as alternativas encontradas tem-se:

- Serviço Web de envio de SMS: empresas na internet que oferecem serviços de envio de mensagens SMS pela Web.
- Módulo: Desenvolver um módulo GSM que pudesse ser integrado à placa NetControl.
- Modem GSM: utilizar um modem GSM para envio de mensagens SMS.

A primeira alternativa, mesmo após o esforço de integrar o sistema a algum tipo de serviço de envio SMS, teria o risco de deixar o sistema dependente uma empresa terceira. Futuramente, se essa empresa parasse de fornecer o serviço, o sistema deixaria de ter esse recurso de envio de mensagem.

A segunda e a terceira alternativa foram as mais aceitáveis, pois deixariam o sistema dependente apenas das operadoras de celular que são as responsáveis pelo envio de SMS. A terceira alternativa era a mais rápida, no entanto, os modens GSM encontrados no mercado encareceriam muito o preço do sistema quando se tornasse produto comercial. A segunda alternativa era mais viável mas exige um esforço maior de desenvolvimento. Até a conclusão deste trabalho a empresa ainda não havia decidido qual alternativa usar e ainda estuda outras soluções para o problema. No entanto, para validação do primeiro protótipo do sistema, foi usado um modem GSM/GPRS da Duodigit modelo TC65 que aceita comandos AT por comunicação serial.

Comandos AT é uma linguagem de comandos executados por textos curtos e tem sua especificação aceita pela maioria dos modems. Logo, como solução, bastou-se fazer a ligação de um canal de comunicação serial RS-232 da placa com o modem para que os comandos fossem enviados.

Ao enviar um SMS os seguintes comandos são enviados ao modem via comunicação serial:

- **AT**: testa conexão com o modem.
- **AT+CMGF=1**: configura formato da mensagem para modo texto.
- **AT+CMGS=<da><CR>message<CTRL-Z>/<ESC>**: envia o SMS, onde “<da>” é o número do celular e “message” a mensagem a ser enviada.

Para consulta desses comandos usou-se um manual do fabricante do módulo existente no modem [17].

Capítulo 6: Integração

Neste capítulo é mostrado como é feita a integração do aplicativo para smartphone com a placa controladora do sistema. Primeiramente é apresentado o serviço de transmissão de dados usado, o TCP. Depois é apresentada a arquitetura de comunicação, o modelo de integração cliente/servidor. Por fim, mostra-se o protocolo de troca de mensagens da aplicação criado pelo autor deste documento para a comunicação entre o aplicativo e a placa.

6.1: Protocolo TCP/IP

O modelo de referência TCP/IP foi definido pela primeira vez por Cerf e Kahn em 1974. Esse modelo surgiu da necessidade do Departamento de Defesa dos Estados Unidos de manter a conexão entre máquinas de origem e destino intactas mesmo que algumas máquinas ou linhas de transmissão intermediárias parassem de funcionar repentinamente [18]. Em outras palavras, problemas de hardware ou perda de dados por carga excessiva na rede não poderiam afetar a comunicação entre duas máquinas.

Este modelo de rede pode ser dividido em quatro camadas: aplicação, transporte, inter-redes e host/rede. A figura 16 mostra uma comparação deste modelo com o modelo OSI [18].

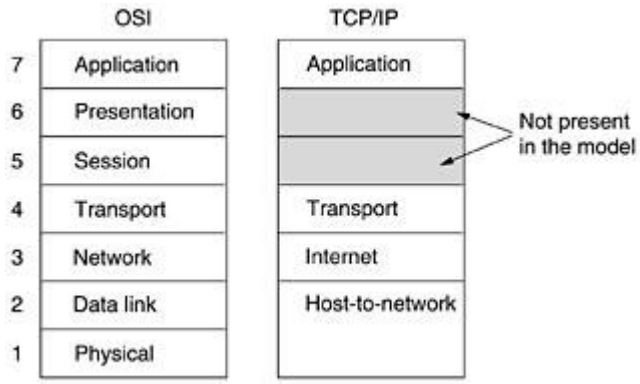

Figura 16 - As quatro camadas do modele TCP/IP e sua comparação com o modelo OSI

As quatro camadas definidas no modelo TCP/IP são:

- **Camada de aplicação:** essa camada está mais próxima do usuário e contém os protocolos de nível mais alto. Aqui ficam os protocolos de comunicação de nível de aplicação como: TELNET, FTP e SMTP que são mostrados na figura 17. Para este sistema foi desenvolvido um protocolo de aplicação que será explicado mais detalhadamente ainda neste capítulo.

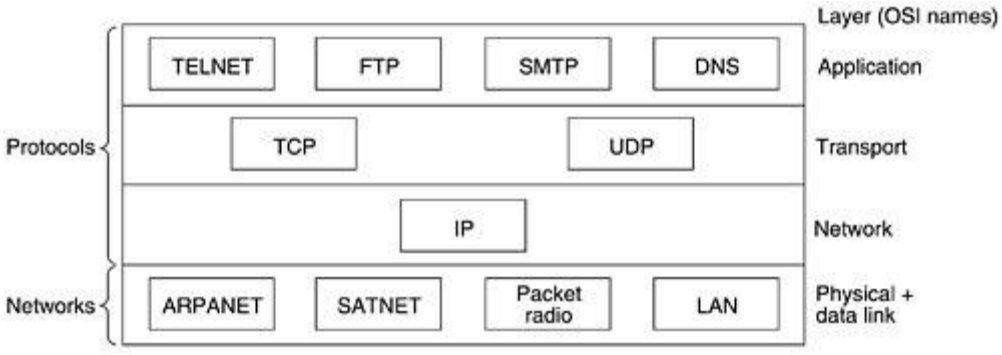

Figura 17 - Protocolos usados em cada camada do modelo TCP/IP

- **Camada de transporte:** a camada de transporte tem como objetivo permitir que as entidades de hosts de origem e destino mantenham uma conversação. Nesta camada foram definidos dois protocolos: o

TCP e o UDP. O TCP (Transmission Control Protocol – protocolo de controle de transmissão) é um protocolo orientado a conexão que garante a entrega confiável de uma cadeia de bytes de uma máquina de origem para uma de destino na rede. Esse protocolo fragmenta as mensagens e o envio pode ocorrer por caminhos diferentes. No destino, a mensagem, que pode chegar fora de ordem, é remontada. O TCP também cuida do controle de fluxo para impedir a sobrecarga de um receptor por um transmissor mais rápido. O UDP (User Datagram Protocol – Protocolo de datagrama do usuário) é um protocolo não orientado a conexão e não confiável. É muito utilizado em aplicações em que a entrega imediata é mais importante que entrega precisa como transmissão de dados de áudio e vídeo.

- **Camada inter-redes:** essa camada tem o papel de permitir que hosts coloquem pacotes em qualquer rede e garantir que os mesmos trafeguem independentes até o destino, podendo percorrer caminhos diferente e até chegarem em ordens diferentes de envio.
- **Camada host/rede:** esta camada não é bem definida dentro do protocolo, exceto o fato de que o host deve conectar à rede usando algum protocolo para que seja possível enviar pacotes IP.

6.2: A interface de socket

Um socket é uma abstração que serve para designar um elo de comunicação entre dois ambientes computacionais. O padrão de interface socket surgiu originalmente no sistema Unix BSD (Berkeley Software Distribution) e hoje é a base para transmissão/recepção (I/O) em redes [19].

Pode-se imaginar socket como um ponto terminal de comunicação que segue as primitivas abrir-ler-escrever-fechar. Esse é o princípio de funcionamento usado no aplicativo para smartphone para comunicar-se com a placa. Como a API do Android possui várias classes do núcleo Java, o uso de sockets para comunicação torna-se bastante simples. A figura 18 mostra a função de envio de mensagem do aplicativo para a placa.

```
public String enviarMensagem(String msg)
{
    String ret = null;
    try
    {
        Socket socket = new Socket();
        InetSocketAddress addr = new InetSocketAddress(enderecoDestino, portaDestino);
        socket.connect(addr, TIMEOUT);

        if (socket.isConnected())
        {
            DataInputStream in = new DataInputStream(socket.getInputStream());
            DataOutputStream out = new DataOutputStream(socket.getOutputStream());

            String outMsg = msg;
            byte[] bufferOut = outMsg.getBytes();
            out.write(bufferOut);

            byte[] bufferIn = new byte[200];
            in.read(bufferIn);

            String inMsg = new String(bufferIn).trim();

            socket.close();

            ret = inMsg;
        }

    }
    catch (NullPointerException ex)
    {
        Log.d("DEBUG", "NullPointerException");
    }
    catch (UnknownHostException ex)
    {
        Log.d("DEBUG", "UnknownHostException");
    }
    catch (IOException ex)
    {
        Log.d("DEBUG", "IOException");
    }

    return ret;
}
```

Figura 18 - Código da função de envio de mensagem do aplicativo

Essa função pertence à classe “Conexao”. Seu funcionamento básico é abrir um socket, conectar ao endereço e porta definidos da placa, enviar a mensagem de parâmetro, esperar a resposta da placa e, por fim, fechar o socket e retornar a resposta recebida da placa.

6.3: A arquitetura cliente/servidor

A arquitetura usada para integrar o sistema, o aplicativo com a placa, foi a cliente/servidor. Nesta arquitetura existe um processo servidor que oferece serviços na rede e um ou mais processos clientes que solicitam serviços do servidor e esperam por respostas. *Um servidor começa a execução antes que a interação comece e (geralmente) continua a aceitar as solicitações e a transmitir respostas sem mesmo terminar. Um cliente é qualquer programa que faz uma solicitação e espera uma resposta. Ele (geralmente) termina depois de usar um servidor durante determinado número de vezes* [19].

As principais características dos clientes são: iniciar os pedidos aos servidores, esperar por respostas e receber as respostas. Geralmente interage diretamente com o usuário final por meio de alguma interface de usuário e é quem utiliza os recursos da rede.

As principais características do servidor são: esperar por solicitações do cliente, atender ao pedido e responder com os dados solicitados. É ele quem fornece os recursos da rede.

Diante dessas características expostas fica evidente que, neste trabalho, o aplicativo se comporta como um cliente e a placa NetControl é o servidor que executa os serviços e fornece as informações do sistema. A figura 19 mostra a arquitetura cliente/servidor aplicada ao sistema.

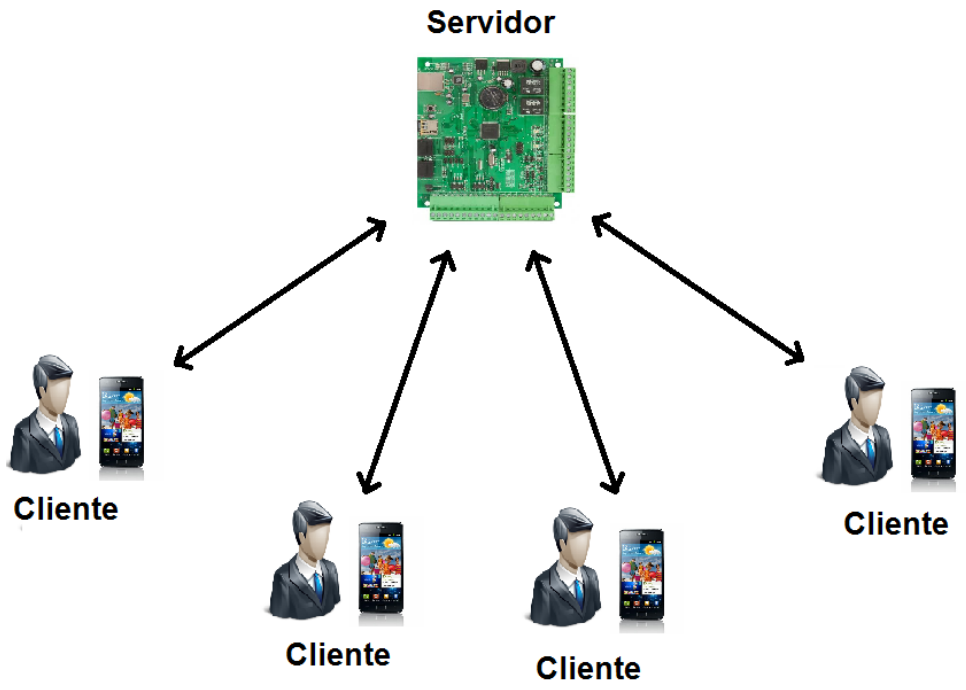

Figura 19 - Arquitetura cliente/servidor

6.4: Protocolo da aplicação

A comunicação do aplicativo com a placa se dá por meio de um protocolo de troca de mensagens. Esse protocolo é simples e foi criado pelo próprio autor deste trabalho.

Este protocolo funciona da seguinte forma, cada mensagem enviada pelo aplicativo é composta por duas partes, uma parte de identificação e outra de dados. A parte de identificação serve para especificar a requisição que o aplicativo solicita a placa. A parte de dados é usada para passar os parâmetros da requisição para a placa. A figura 20 mostra um esquemático de como é formada a mensagem.

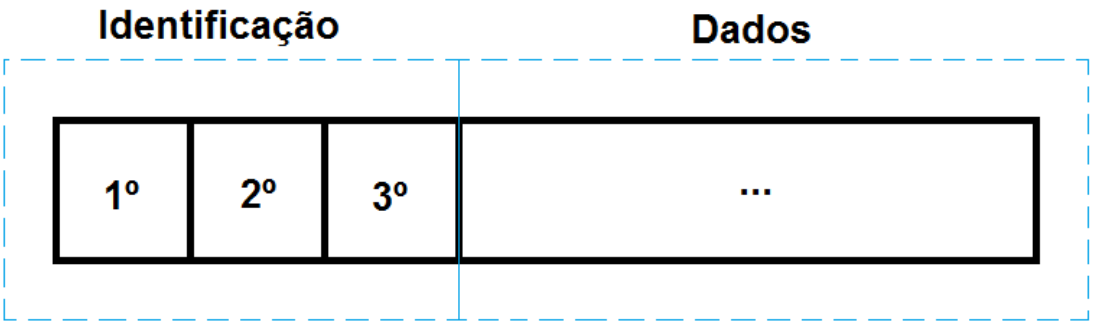

Figura 20 - Estrutura da mensagem do protocolo de comunicação do aplicativo com a placa

A parte de identificação da mensagem é composta por 3 bytes, cada um é representado por um caractere de identificação:

- Primeiro byte: caractere que representa o dispositivo do sistema do qual se quer solicitar o serviço. A tabela abaixo mostra os bytes de identificação do dispositivo.

Tabela 6 - Caracteres de identificação de cada dispositivo do sistema

Caractere	Dispositivo
p	Porta
e	Portão eletrônico
a	Alarme
b	Botão de pânico
l	Login

- Segundo byte: caractere que representa a função (ou serviço) do dispositivo. Como cada dispositivo tem recursos diferentes, a seguir são mostradas as tabelas de caracteres com os respectivos recursos solicitados.
- Terceiro byte: é um campo reservado. Atualmente ele é usado como um código de identificação da porta, mas não há necessidade já que

existe apenas uma porta no sistema. Pode ser usado futuramente para outro fim.

As quatro tabelas a seguir mostram as requisições que cada dispositivo tem e quais caracteres devem ser colocados na mensagem para fazê-las.

Tabela 7 - Requisições da porta

Caractere	Requisição	Argumento
p	Solicita que a placa destranque a porta.	-
s	Solicita o estado da porta: aberta ou fechada.	-
g	Solicita o tipo de fechadura da porta.	-
x	Obtém o tempo de abertura da fechadura da porta.	-
y	Grava um novo tempo de abertura para a fechadura da porta.	Dois bytes representando o tempo.

Tabela 8 - Requisições do portão eletrônico

Caractere	Requisição	Argumento
p	Solicita abertura do portão eletrônico.	-

Tabela 9 - Requisições do alarme

Caractere	Requisição	Argumento
a	Ativa o alarme.	-
d	Desativa o alarme.	-
s	Obtém o estado do alarme (ativado, desativado ou disparado).	-
m	Obtém a mensagem de SMS de alarme.	-
g	Grava a nova mensagem de SMS de alarme.	Mensagem
i	Insere novo contato de celular para envio de SMS de alarme.	nome + # + número
h	Obtém a lista de celulares configurados para envio de SMS de alarme.	-
e	Deleta um cadastro de celular para envio de SMS de alarme.	Id do cadastro
c	Edita um contato de SMS de alarme.	Id + nome + numero

Tabela 10 - Requisições do botão de pânico

Caractere	Requisição	Argumento
e	Envia o SMS de pânico para os celulares cadastrados.	-
m	Solicita a mensagem SMS de pânico.	-
g	Grava mensagem SMS de pânico.	Mensagem
h	Obtém lista de celulares cadastrados de envio de SMS de pânico.	-
d	Exclui cadastro de celular para envio de SMS de pânico.	Id do cadastro
i	Inseri novo cadastro de celular para envio de SMS de pânico.	nome + # + número
c	Edita cadastro de celular para envio de SMS de pânico.	Id + nome + numero

Para exemplificar, caso o aplicativo queira solicitar à placa a mensagem SMS de pânico cadastrada no sistema, a mensagem a ser enviada será “pm”. O primeiro caractere, segundo a tabela 6, é o “p” para o botão de pânico. O segundo caractere, segundo a tabela 10, é o “m” para solicitar o texto da mensagem SMS. Após o envio dessa mensagem pelo aplicativo, a placa retorna o texto da mensagem gravada em sua memória.

Todas as rotinas que executam essas requisições são definidas nos arquivos mencionados na seção 5.5.1.

Capítulo 7: Testes e Resultados

Este capítulo apresenta e discute os testes e resultados do sistema de segurança residencial integrado ao aplicativo para smartphones desenvolvido neste trabalho. Até a escrita deste documento, não foi possível testar todas as funcionalidades desenvolvidas, mas as que foram testadas apresentaram bons resultados. O sistema encontra-se instalado internamente na empresa e está em fase de testes.

7.1: Instalação

A primeira versão do sistema foi instalada na porta da sala do setor de desenvolvimento da empresa. Pois a mesma já possuía uma fechadura eletromagnética, da Automatiza, instalada e funcionando com outro sistema de controle de acesso da empresa.

Primeiramente o firmware foi gravado em uma placa NetControl. Em seguida a mesma foi fixada em uma caixa de suporte presa na parede superior à porta. Após isso, foi configurada uma rede sem fio em um roteador wi-fi e ligou-se o mesmo a placa por meio de um cabo ethernet.

O acionamento da fechadura eletromagnética é simples. A fechadura, quando alimentada, é magnetizada e tranca a porta, caso contrário ela destranca. Logo, apenas liga-se um dos cabos de alimentação da fechadura no relé da placa e então a mesma passa a ser acionada quando o relé fecha o circuito. Como o estado normal da porta é trancado, ligou-se um fio no terminal NF (normalmente fechado) e outro no comum. A abertura do portão não foi possível de ser testada, pois não foi possível montar um cenário de testes com um portão eletrônico até a execução deste trabalho. Mas o princípio de instalação e funcionamento é semelhante ao da porta.

Para o teste do alarme foi adquirido um sensor de presença. O sensor de presença possui dois fios de ligação. Um relé mantém esses dois fios em contato e quando o sensor é sensibilizado abre-se a ligação. Então, ligou-se

um dos fios do sensor na referência da placa e o outro na entrada ALM (alarme), que é a entrada que o firmware faz a leitura do sensor.

O modem GSM/GPRS também foi instalado para o envio de mensagens SMS. O mesmo foi ligado ao canal 1 de comunicação serial da placa.

O aplicativo foi instalado nos smartphones de alguns funcionários da empresa. Para efetuar a instalação, basta copiar o instalador (arquivo ".apk" gerado após a compilação do código) para um diretório do smartphone e instalar o aplicativo. Feita a instalação conecta-se o wi-fi do celular na rede e configura o IP e porta de comunicação com a placa.

7.2: Avaliação do sistema

Após a instalação do sistema, o mesmo passou a ser usado pelos usuários cadastrados da empresa. Até o término deste trabalho o sistema estava em funcionamento havia três semanas.

Após esse período de testes do sistema, algumas considerações podem ser feitas quanto ao cumprimento dos requisitos do projeto e ao desempenho do sistema:

- A interface do aplicativo exibe, de fato, o estado da porta (trancada ou destrancada) e do alarme (ativado, desativado e disparado). A atualização é feita a cada 3 s.
- Pode-se cadastrar, editar e excluir usuários do sistema e celulares para envio de mensagem SMS de alarme e pânico.
- A porta é destrancada por meio do aplicativo.
- O alarme é ativado e desativado pelo aplicativo.
- Quando o alarme está ativado e o sensor é sensibilizado a mensagem de SMS é enviada a todos os celulares cadastrados.
- Quando o botão de pânico é acionado a mensagem de SMS é enviada a todos os celulares cadastrados.

O registro de eventos, que era um requisito que estava inicialmente no escopo deste trabalho, foi retirado e, portanto, não foi desenvolvido. No entanto, ainda é uma melhoria futura do sistema e será implementado.

Alguns problemas também apareceram na fase de testes como a lentidão de resposta da interface gráfica. A mesma apresentava um feedback lento para o usuário quando a rede estava lenta. Essa lentidão deve-se ao fato de o aplicativo executar as requisições à placa na mesma thread da interface gráfica. Esse problema foi posteriormente corrigido com o uso de uma nova thread para efetuar requisições e então liberar a interface gráfica.

O sistema continua em desenvolvimento e ainda vai passar por testes mais rigorosos até que o seja lançado como produto para cliente.

Capítulo 8: Conclusões e Perspectivas

Neste trabalho foi apresentado o desenvolvimento de um sistema de segurança para ambientes residenciais. O diferencial desse projeto é a integração do sistema com uma interface para smartphone possibilitando ao usuário a interação remota com o sistema e a mobilidade com relação ao mesmo.

O trabalho apresentou bons resultados para a empresa que dará continuidade ao projeto até que o mesmo possa ser lançado no mercado como um produto.

Muitas melhorias ainda devem ser feitas no sistema para que isso aconteça. Uma delas é no próprio hardware do sistema, pois é preciso desenvolver um hardware que possua um módulo para envio de mensagens SMS, dado que um modem não é viável economicamente para esse fim. No firmware ainda deve ser implementado um sistema de banco de dados para registro de eventos, o que pode ser implementado no cartão SD existente na placa NetControl.

Outra melhoria é o próprio desenvolvimento de versões do aplicativo para outras plataformas móveis como o iOS e o Windows Phone. Como o servidor, a placa controladora, do sistema está pronto, o desenvolvimento das outras versões do aplicativo será facilitado. Pois basta desenvolver apenas o cliente seguindo o protocolo de comunicação criado.

Este projeto teve grande contribuição para a formação pessoal e acadêmica do autor deste documento. A convivência diária em um ambiente profissional proporcionou muitas experiências importantes que não podem ser vistas somente na graduação. Do ponto de vista acadêmico, muito se agregou na bagagem profissional do seu autor. O desenvolvimento de um firmware em linguagem C e um aplicativo em linguagem Java de forma concorrente proporcionou uma experiência muito interessante pela diferença de estrutura e nível de abstração de cada linguagem. Esse trabalho serviu para validar todo o

conhecimento adquirido durante o curso de Engenharia de Controle e Automação da UFSC.

Bibliografia:

[1] ALVES, José. **Casas Inteligentes**. Centro Atlântico. Portugal. 2003.

[2] MARIMOTO, Carlos Eduardo. **Smartphones, guia prático**. Sul Editores. Brasil. 2009.

[3] IDC CORPORATE USA. **Nearly 1 Billion Smart Connected Devices Shipped in 2011 with Shipments Expected to Double by 2016.** Acesso em junho de 2012. Disponível em: http://www.idc.com/getdoc.jsp?containerId=prUS23398412.

[4] IDC CORPORATE USA. **China to Become the Largest Market for Smartphones in 2012 with Brazil and India Forecast to Join the Top 5 Country-Level Markets by 2016.** Acesso em fevereiro de 2012. Disponível em: http://www.idc.com/getdoc.jsp?containerId=prUS23381112.

[5] IDC CORPORATE USA. **Android - and iOS- Powered Smartphones Expand Their Share of the Market in the First Quarter.** Acesso em junho de 2012. Disponível em: http://www.idc.com/getdoc.jsp?containerId=prUS23503312.

[6] GARTNER. **Gartner Says Android to Command Nearly Half of Worldwide Smartphone Operating System Market by Year-End 2012.** Acesso em fevereiro de 2012. Disponível em: http://www.gartner.com/it/page.jsp?id=1622614.

[7] APPLE DEVELOPER. **Designed, build, code, innovate**. Acesso em junho de 2012. Disponível em: https://developer.apple.com/

[8] ANDROID DEVELOPER**. API Guides**. Acesso em junho de 2012. Disponível em: http://developer.android.com/guide/components/index.html

[9] BGR. **Android Market surpasses 500,000 published apps**. Acesso em junho de 2012. Disponível em: http://www.bgr.com/2011/10/21/android-market-surpasses-500000-published-apps.

[10] MSDN. **Installing the Windows Phone SDK**. Acesso em junho de 2012. Disponível em: http://msdn.microsoft.com/pt-BR/library/ff402530(v=vs.92).aspx#BKMK_SysReqs.

[11] BLACKBERRY DEVELOPER. **Documentation**. Acesso em junho de 2012. Disponível em: https://developer.blackberry.com/java/documentation.

[12] NOKIA DEVELOPER. **Symbian platform**. Acesso em junho de 2012. Disponível em: http://www.developer.nokia.com/Devices/Symbian/.

[13] ANDROID DEVELOPERS. **Appications Fundamentals**. Acesso em junho de 2012. Disponível em: http://developer.android.com/guide/components/fundamentals.html.

[14] WAZLAWICK, Raul Sidnei. **Análise e Projeto de Sistemas de Informação Orientados a Objetos**. Elsevier. Brasil.2004.

[15] FREERTOS. **About FreeRTOS**. Acesso em junho de 2012. Disponível em: http://www.freertos.org/RTOS.html.

[16] DUNKELS, Adam. **The uIP Embedded TCP/IP Stacks**. Swedish Institute of Computer Science. 2006.

[17] CINTERIUM WIRELESS MODULES. **TC65 AT Command Set**. Cinterium Wireless Modules. Alemanha. 2008.

[18] TANENBAUM, Andrew S. . **Redes de Computadores**. Editora Campus. 2003. Brasil.

[19] COMER, Douglas E. . **Interligação em rede TCP/IP**. Editora Campus. 1998. Brasil.