UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

CONCEPÇÃO DE UMA SOLUÇÃO DE AMBIENTE INTELIGENTE AUTOMÁTICA PARA AUXILIAR NA MEDICAÇÃO DE PACIENTES

CLAUDIO EDUARDO MARQUES GOMES

MANAUS-AM
2013

UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

CLAUDIO EDUARDO MARQUES GOMES

CONCEPÇÃO DE UMA SOLUÇÃO DE AMBIENTE INTELIGENTE AUTOMÁTICA PARA AUXILIAR NA MEDICAÇÃO DE PACIENTES

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica na Área de concentração: Controle e Automação de Sistemas.

Orientador:

Prof. Dr. –Ing. Vicente Ferreira de Lucena Júnior

MANAUS
2013

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

M357c

Marques Gomes, Claudio Eduardo
Concepção de uma Solução de Ambiente Inteligente Automática para Auxiliar na Medicação de Pacientes / Claudio Eduardo Marques Gomes. 2013
170 f.: il. color; 29,7 cm.

Orientador: Prof. Dr. –Ing. Vicente Ferreira de Lucena Júnior
Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal do Amazonas.

1. desenvolvimento centrado no usuário. 2. armário de medicamento inteligente. 3. sistemas automatizados. 4. adesão à medicação. 5. RFID. I. Lucena Júnior, Prof. Dr. –Ing. Vicente Ferreira de II. Universidade Federal do Amazonas III. Título

UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

CONCEPT OF AN AMBIENT ASSISTED LIVING SOLUTION TO AID IN PATIENT MEDICATION

CLAUDIO EDUARDO MARQUES GOMES

MANAUS-AM
2013

UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

CLAUDIO EDUARDO MARQUES GOMES

CONCEPT OF NA AMBIENT ASSISTED LIVING SOLUTION TO AID IN PATIENT MEDICATION

Master Thesis presented for the Master in Electrical Engineering Field of Control and Automation Systems for the "Programa de Pós - Graduação em Engenharia Elétrica" at Federal University of Amazonas.

Supervisor:

Prof. Dr. –Ing. Vicente Ferreira de Lucena Júnior

MANAUS
2013

CLAUDIO EDUARDO MARQUES GOMES

CONCEPÇÃO DE UMA SOLUÇÃO DE AMBIENTE INTELIGENTE AUTOMÁTICA PARA AUXILIAR NA MEDICAÇÃO DE PACIENTES

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica na área de concentração Controle e Automação de Sistemas.

Aprovado em 21 de Junho de 2013.

BANCA EXAMINADORA

Prof. Dr.- Ing Vicente Ferreira de Lucena Junior, Presidente

Universidade Federal do Amazonas- UFAM

Prof. Dr. Waldir Sabino da Silva Junior, Membro

Universidade Federal do Amazonas- UFAM

Prof. Dr. Eduardo James Pereira Souto, Membro

Universidade Federal do Amazonas- UFAM

Dedico este trabalho à minha família pelo apoio à minha educação, mas especialmente os meus pais, que trabalharam duro para que eu tivesse as condições necessárias para continuar estudando e para alcançar os meus sonhos, e ao Professor Vicente Ferreira de Lucena Júnior, meu orientador, que acreditou no meu potencial, tornando possível o meu intercâmbio na Alemanha, me incentivando a dar sempre um passo à frente.

Agradecimentos

Primeiramente, gostaria de agradecer à Deus por ter sempre me guiado durante meus estudos.

Agradeço meu orientador, Prof.-Ing Dr. Vicente F. de Lucena Júnior, por ter me orientado de forma apropriada para que eu pudesse concluir minha tese de mestrado tanto na Alemanha quanto no Brasil.

Gostaria de agradecer meus colegas do CETELI, Orlewilson Bentes, Ricardo Erikson, Vandermi Silva, Water Simões, Mauro Lopes e Victor Lauria, que me deram dicas e orientaram minha pesquisa, bem como me ajudaram com minhas publicações.

Agradeço ao Fabrício de Souza Farias, que, mesmo em outro continente, também ajudou na minha tese e deu dicas para se ter sucesso nas publicações. A lista de amigos, que também me ajudaram em outro país, inclui: César Rocha, Elôa Jaqueline, Ana Franco, Marcel Maier, Leandro Camargo, Larissa Petri, Marcelo Kipper, Christian Lange, William Padilha, Renan Baima and Jeremy Yam.

Gostaria de agradecer à Suellen Santana, que sempre esteve ao meu lado, mesmo antes da minha qualificação, durante os momentos mais difíceis.

Agradeço aos Profs. José Luiz de Souza Pio e Waldir Sabino pro orientar minha pesquisa quando ela ainda estava no início.

Essa dissertação foi desenvolvida em parceria com o "Institut für Automatisierungs- und Softwaretechnik" (IAS) na Alemanha. Dessa forma, gostaria de agradecer ao Prof. Dr.-Ing. Dr. h. c. Peter Göhner, Dr.-Ing. Nasser Jazdi, Mr. Reiner Villing e aos funcionários pela ajuda prestada.

Agradeço a Srta. Ana Luísa Almada, que foi um dos médicos que ajudaram essa dissertação com informações preciosas sobre o processo de medicação baseado na experiência dela na área.

Também gostaria de agradecer minha família: meu pai Claudio Coelho, minha mãe Márcia, minha tia Hilda, tio Denni, e meus irmãos, Suelen, Sabrina e Júnior.

Essa dissertação de mestrado é muito importante para mim, já que ela representa um período da minha vida no qual pude aprender muito sobre o significado de pesquisa e acumulei bastante conhecimento prático de Engenharia na Alemanha. Dessa forma, gostaria de agradecer também a todas as outras pessoas que contribuíram comigo de alguma forma.

"I refuse to walk away from the fight.
It's my nature. To go right to the end."

Ayrton Senna (3 vezes campeão mundial de F1).

Zusammenfassung

Benutzer Orientierte Entwicklung ist ein neuer Trend in der Entwicklung automatisierter Systeme. In diesem Paradigma spielt der Nutzer eine zentrale Rolle während des gesamten Entwicklungsprozesses. Die Bedürfnisse, Wünsche, Einschränkungen und Präferenzen der Nutzer werden gründlich analysiert und systematisch in das automatisierte System integriert. Dies beeinflusst die Systemarchitektur und seine Benutzeroberfläche bereits vom Beginn des Entwicklungsprozesses. Das Ergebnis ist ein Produkt, das auf seinen Endnutzer zugeschnitten ist. Benutzbarkeit und Barrierefreiheit sind zwei wichtige Aspekte solcher Produkte, die in der Literatur häufig angesprochen werden. Sie definieren den Grad, wie einfach und intuitiv ein Produkt von so vielen Nutzern wie möglich verwendet werden kann.

Am Institut für Automatisierungs-und Softwaretechnik (IAS) wurde nach dem Paradigma der benutzerorientierten Entwicklung ein intelligenter Arzneischrank entwickelt. Der intelligente Arzneischrank verwaltet die Medizineinnahmen von seinen Nutzern, indem er ihnen Anweisungen und/oder Warnhinweise auf ihre Medikamente gibt. In dieser Master Thesis wurde der intelligente Arzneischrank um einige Funktionen und mehr Stabilität erweitert. Verwaltung von Rezepten, Warnung bei unerlaubtem Zugriff, Überwachung der Tagesdosis, mehrsprachige Sprachausgabe und Internet-Schnittstelle sind Errungenschaften dieser Arbeit.

Keywords: benutzer orientierte entwicklung, intelligenter arschneischrank, automatisierte systeme, medikationsfesthalten, RFID

Abstract

User centered development has become a new trend in development of industrial automation systems. In this paradigm the user plays a central role during the whole development process. The users' needs, wishes, restrictions and preferences are being thoroughly analyzed and systematically integrated into the automation system. This affects the system architecture and its user interface already from the beginning of the development process. The result is a product, which is tailored to its end users. Usability and accessibility are two important aspects of such products, which are often addressed in the literature. They define the degree to which a product can be easily and intuitively used by as many users as possible.

At the Institute of Industrial Automation and Software Engineering (IAS) an intelligent medicine cabinet has been developed, according to the user centered development paradigm. The intelligent medicine cabinet administers the medicine takings of its users, by giving them instructions and/or warnings on their medications. In this master thesis, the medicine cabinet has been extended to numerous functions and more stability. Administration of prescriptions, warning on unauthorized access, monitoring of allowed daily dosage, multilingual speech outputs and internet interface are among the achievements of this work.

Keywords: user oriented development, intelligent medicine cabinet, automation systems, medication adherence, RFID

Resumo

O desenvolvimento centrado no usuário tem se tornado uma nova tendência no desenvolvimento de sistemas de automação industriais. Nesse paradigma, o usuário tem um papel central durante todo o processo de desenvolvimento. As suas necessidades, desejos, limitações e preferências estão sendo analisadas exaustivamente e integradas de forma sistemática no sistema de automação. Isso afeta a arquitetura do sistema e sua interface de usuário já desde o início do processo de desenvolvimento. O resultado é um produto, que é feito sob medida para os usuários finais. Usabilidade e acessibilidade são dois aspectos importantes de tais produtos, as quais são frequentemente abordadas na literatura. Elas definem até que grau um produto pode ser fácil e intuitivamente usadas pela maior quantidade de usuários possíveis.

No Instituto de Automação Industrial e Engenharia de Software (IAS) em Stuttgart, um armário de medicamento inteligente foi concebido seguindo os preceitos do paradigma de desenvolvimento centrado no usuário. O armário de medicamento inteligente administra o consumo de medicamentos dos seus usuários, dando a eles instruções e/ou advertências quanto às medicações deles. Nessa tese de mestrado, o armário de medicamento foi estendido com várias funcionalidades e com maior estabilidade. Administração de prescrições, avisos quanto ao acesso não autorizado, monitoramento da dose diária máxima recomendada, instruções em diversos idiomas e uma interface web estão entre alguns dos objetivos alcançados nesse trabalho.

Palavras-chave: desenvolvimento centrado no usuário, armário de medicamento inteligente, sistemas automatizados, adesão à medicação, RFID

List of Figures

List of Tables

List of Abbreviations

ASCII	***A**merican **S**tandard **C**ode for **I**nformation **I**nterchange*
API	***A**pplication **P**rogramming **I**nterface*
CRC16	***C**yclic **R**edundancy **C**ode 16*
dsPIC	***D**igital **S**ignal **P**rogrammable **I**nterface **C**ontroller*
eSATA	***E**xternal **S**erial **A**dvanced **T**echnology **A**ttachment*
FAQ	***F**requently **A**sked **Q**uestions*
HDMI	***H**igh **D**efinition **M**ultimedia **I**nterface*
HTTP	***H**yper **T**ext **T**ransfer **P**rotocol*
ID	***Id**entification*
JDBC	***J**ava **D**ata**b**ase **C**onnectivity*
JSON	***J**ava **S**cript **O**bject **N**otation*
LED	***L**ight **E**mitting **D**iode*
MP3	***MP**EG-1/2 Audio Layer **3***
MySQL	***My S**tructured **Q**uery **L**anguage*
NFC	***N**ear **F**ield **C**ommunication*
PHP	**PHP**: ***H**ypertext **P**reprocessor*
PIC	***P**rogrammable **I**nterface **C**ontroller*
SAPI	***S**peech **A**pplication **P**rogramming **I**nterface*
SATA	***S**erial **A**dvanced **T**echnology **A**ttachment*
SQL	***S**tructured **Q**uery **L**anguage*
TTS	***T**ext **T**o **S**peech*
UML	***U**nified **M**odeling **L**anguage*
USB	***U**niversal **S**erial **B**us*
VGA	***V**isual **G**raphic **A**rray*
W3C	***W**orld **W**ide **W**eb **C**onsortium*

Terminology

Hashcode A hash code is a numerical value generated by an algorithm or subroutine that maps large data sets of variable length to smaller data sets of a fixed length. The name of a person with four words, for example, could be mapped to a single integer value. Hashcode is usually used to associate large data sets to a key value, which can be used on a table as an index.

USB USB stands for Universal Serial Bus. It was developed in the 90's to standardize the connection of computer peripherals, both for communication and power supply. It can both transfer enable the communication, as well as provide power supply to certain devices.

HTTP request An HTTP request is a request that uses Hyper Text Transfer Protocol commands. These commands work in a client-host style. The client sends requests to the host, which process it and sends the result to the client. The result contains information related to the request made by the client

Framework A framework is a piece of software that provides general functionalities to the user, which has only to change some part of its code to meet his needs. A framework has universal use to what it is proposed and may be reused as a working solution, only needing some adaptations.

UML It is a standardized modeling language to be used specially for object-oriented applications. It provides well defined rules to create visual models, which may be easily understood to have an overview of the application solution.

AIDC It stands for "automatic identification and data capture" and it is a study filed that applies methods to identify objects automatically, collecting data about them, and sending that data directly into computer systems, in other words, there is no need of human intervention on such process.

SOAP It stands for "simple object access protocol". It is a protocol that was design to

provide interoperability between systems, independently of how they are implemented. It is one of the implementations of web services and it relies on XML for its message format, allowing structured information exchange. For its communication, it relies on application layers protocols, such as HTTP.

XML Standing for "extensible markup language", it defines a set of rules to format date in a structured way that is both readable for humans and machines. It was designed to provide simplicity and usability, which are desirable features for web services for the representation of arbitrary data structures.

WSDL It stands for "web service description language". It is an XML file, therefore human-readable, that contains the description of a web service, showing how the service can be called, what its parameters are, which values they accept, and what type of answer each operation returns.

Table of Contents

Resumo Estendido

Devido à natureza dessa tese de mestrado – concebida na Universidade de Sttugart, Alemanha – o texto original foi feito em Inglês. Dessa forma, esse capítulo tem como objetivo mostrar uma visão geral do trabalho feito, funcionando como um espelho (resumo) do conteúdo em Inglês dos capítulos seguintes. Para facilitar o entendimento, os tópicos desse capítulo têm uma estrutura semelhante ao restante do documento. Para maiores detalhes, consultar os capítulos em Inglês referentes aos sub-tópicos seguintes.

Introdução e motivação

O mundo tem visto melhorias significativas na área de assistência médica (*healthcare*), aumentando a expectativa de vida global e disponibilizando remédios cada vez mais eficientes no mercado para tratar uma variedade de problemas de saúde. Todavia, tal disponibilidade trouxe também efeitos colaterais: as pessoas estão tomando cada vez mais remédios por conta própria, sem ao menos consultar um médico.

Alguns jornais chegaram a reportar esse problema com as seguintes manchetes: “Pais processam depois que filho cospe sangue e morre ao tomar medicamento de *recall*”; “Garota morre após tomar uma combinação letal de dois medicamentos comuns para alergia e resfriado”; “Uma avó que sofria de dores crônicas nas costas esqueceu que já tinha tomado a dose diária de pílulas e acabou em overdose”. O número de mortes induzidas por medicamentos nos EUA, por exemplo, chegou a ultrapassar o número de fatalidades no trânsito em 2011.

É visível a necessidade de soluções *healthcare* para melhorar a adesão à medicação. A concepção dessas soluções, no entanto, não é trivial devido à necessidade de se fazer sistemas com maior aceitação e usabilidade. O desafio de se desenvolver sistemas com tais características deve-se ao fato de que capturar todas as necessidades e limitações dos usuários é uma tarefa complexa. Uma área que lida com isso é a “*Ambient Assisted Living*” (AAL). Ela foca em ajudar os idosos a terem uma vida mais independente, podendo desempenhar suas atividades diárias de forma mais autônoma.

Tomar medicamentos é uma atividade bastante comum em um contexto de AAL, cujo tema é o foco dessa tese de mestrado. De forma geral, idosos tomam mais medicamentos e estão mais propensas do que os jovens a esquecerem de tomá-los na hora certa ou têm dificuldade ao ler informações médicas sobre medicamentos. Além disso, o mundo está ficando cada vez mais velho. Uma pesquisa (ver seção 2.1) revelou que a população mundial de idosos está crescendo a uma taxa muito rápida. A previsão é de que a população com 60 anos ou mais irá aumentar em mais de 50% nas próximas quatro décadas.

Uma população mais velha significa uma maior necessidade de assistência médica e, como a medicação desempenha um papel importante no cotidiano dos idosos, há uma visível necessidade de soluções para o problema da falta de adesão (comprometimento) à medicação nesse contexto AAL. Ainda não há soluções consolidadas que visem o monitoramento e controle do comprometimento do usuário à medicação. Apesar de a informatização de aplicações *healthcare* oferecer uma abordagem melhor e mais fácil para prover orientação médica e permitir captura de dados, a forma como ela é provida para o usuário deve ser feita de forma bem estruturada.

Para ajudar o usuário a tomar medicamentos, poderia se pensar em um simples dispositivo de alarme, através do qual um alerta é gerado na hora certa de se tomar o remédio. Todavia, esse tipo de gerenciamento ainda permitiria falhas, tal como a necessidade de se programar o dispositivo a cada tomada de medicamento. Além disso, não haveria uma interface através da qual o médico, ou cuidadora, poderia acessar tal informação. Dessa forma, a única forma que o médico tem como obter informação do usuário é através do diálogo, em outras palavras, pode haver uma avaliação errônea por parte do médico com base nos dados providos pelo paciente.

Outra falha muito comum na área está relacionada com a prescrição. No Brasil, por exemplo, a única forma através da qual o usuário pode saber quais os medicamentos que devem ser tomados (assim como o momento adequado e dosagem) é através de uma receita que normalmente é escrito à mão ou em um papel impresso. O problema com a versão manuscrita é que o usuário pode não entender claramente as especificações médicas e tomar uma ação prejudicial para o seu tratamento. Quanto à versão impressa, o usuário pode facilmente perdê-la, sem ter uma maneira de recuperar os dados. Na verdade, este é um problema comum para os dois tipos de prescrição.

Os problemas mencionados acima mostram que uma ajuda natural que tenha como objetivo a aceitação máxima por parte do usuário seria uma abordagem apropriada para

aumentar a adesão à medicação. Tal exigência, no entanto, traz muitos desafios em determinar que tipo de auxílio deve ser fornecido (por exemplo, confirmação do consumo pelo usuário ou a detecção automática do mesmo). Todos estes desafios servem como uma motivação para o trabalho desenvolvido nesta dissertação de mestrado.

O objetivo principal dessa dissertação é investigar a concepção de e colocar em execução um sistema em um contexto *Ambient Assisted Living*, que implemente uma solução eletrônica *healthcare*, através do desenvolvimento centrado nas necessidades da medicação do usuário e na realização de um protótipo de armário de medicamentos inteligente. O protótipo do armário de medicamentos inteligente proposto nesse trabalho foi projeto para ser tão simples como a utilização de um armário de remédios comum, aumentando sua usabilidade em um contexto AAL.

Como objetivos específicos, os seguintes tópicos foram comtemplados: investigar e avaliar as características fortes e fracas de sistemas existentes; propor um conceito baseado no estado da arte; avaliar como sistemas *healthcare* existentes poderiam melhorar o uso de informações coletadas dos usuários, medicamentos e prescrições; baseado em sistemas existentes, definir uma solução com mais casos de uso, tentando cobrir a figura do médico no apoio e/ou monitoramento da medicação do paciente; aplicar técnicas de automação centrada no usuário na tentativa de diminuir a possibilidade de falha do sistema, dada a importância solução para a saúde do usuário; definir e desenvolver uma solução eletrônica *healthcare* com uma interface amigável para aumentar a acessibilidade e usabilidade para o processo de medicação; o protótipo proposto deve servir de base para aplicar os conceitos desta tese de mestrado tendo como objetivo adesão à medicação.

Tecnologias relevantes

Radio Frequency Identification (RFID) faz parte de uma classe de tecnologias cujo principal propósito é identificação automática e captura de dados (AIDC). Essa tecnologia é composta de dois componentes principais, através dos quais é possível fazer a identificação de objetos ou pessoas: um leitor e uma *tag*. Dentre os diversos métodos de identificação, o mais comum é armazenar na *tag* sua ID, que identifica o objeto de forma única, sendo possível também armazenar os dados relacionados. Uma *tag* (*transponder)* de medicamento, por

exemplo, poderia armazenar um número de identificação e dados relacionados, tais como o nome do medicamento, dada de validade, quantidade de pílulas e substâncias ativas.

A identificação dos objetos é feita da seguinte forma: a conexão entre a *tag* e o leitor é estabelecida de forma remota através de ondas de rádio e os dados, nessa forma de onda, são interpretados pelo leitor e convertidos para um formato digital, permitindo que a aplicação lide com a informação de acordo com seu uso. Apesar de as *tags* serem feitas em diferentes formatos e materiais de encapsulamento, há somente três classes de *transponders*: ativas, que tem fonte de energia própria e maior alcance; semi-passivas, com energia própria para o circuito interno, mas usa energia do leitor para transmitir os dados; e a passiva, que depende completamente do leitor para transmitir os dados e tem um menor alcance. Para uma aplicação de um armário de medicamento, as passivas são suficientes já que não é necessário um longo alcance, reduzindo também o custo da solução.

Outra tecnologia que também faz parte do grupo AIDC é o código de barras, que consiste em um padrão de barras (listras) pretas e brancas para codificar informação. Na sua forma mais simples, as primeiras listras, que representam o início da leitura, são, respectivamente, preta, branca e preta. O leitor as usa para saber o padrão de espaçamento entre as listras e assim poder decodificar o restante da informação que possui uma barra preta no fim para delimitar onde a leitura deve parar. Como o leitor (*scanner*) precisa receber os sinais elétricos provenientes da reflexão da luz das listras, somente um código de barras pode ser lido por vez. Há duas características dessa tecnologia que devem ser consideradas: a primeira refere-se ao fato de objetos não serem identificados de forma única, já que o número de identificação representa somente classes de objetos/produtos; e a segunda refere-se ao seu custo extremamente baixo.

Uma terceira tecnologia no grupo AIDC que poderia ser usada para o armário de medicamento é o NFC (*Near Field Communication*). De forma geral, NFC é uma tecnologia que permite comunicação sem fio entre dois dispositivos através da proximidade deles, sem a necessidade de configurar uma conexão. Em outras palavras, assim que os dispositivos ficam um ao alcance do outro, a conexão é estabelecida automaticamente e dados podem ser transferidos.

Essa tecnologia foi inicialmente derivada de duas normas que padronizam o RFID, mas um ano após sua criação já possuía sua própria norma. NFC possui dois tipos de dispositivos que o diferenciam do RFID padrão: o iniciador e o *target*. O primeiro é responsável por começar a comunicação e controlar a transferência de dados. Já o segundo,

responde às requisições feitas pelo iniciador. Como NFC faz parte da tecnologia RFID, ela também possui as transmissões passiva e ativa mencionadas acima. Por outro lado, essa tecnologia possui uma característica na qual ela se sobressai: segurança. Além de permitir conexões somente entre dispositivos que estejam no máximo 10 cm de distância um do outro, tal conexão só pode ser estabelecida em um ângulo específico, já que o campo eletromagnético é muito fraco. A tabela abaixo mostra uma comparação entre três tecnologias.

Tabela 1: Comparativo entre código de barras, RFID e NFC

	Código de barras	RFID	NFC
Linha de visada	Necessária	Necessária	Desnecessária
Alcance	De poucos centímetros até alguns metros	Até 100 m com *tags* ativas	Até 10 cm
Taxa de leitura	Uma por vez	Até várias por vez	Uma *tag* por vez
Identificação	Identifica o tipo de item (não de forma única)	Cada item identificado de forma única	Leitura/escrita, ponto-a-ponto e emulação de cartão
Leitura/Escrita	Somente leitura	Ambos	Ambos
Tecnologia	Óptica (*laser*)	Frequência de rádio	Frequência de rádio
Interferência	Não pode ser lido se obstruído	Metal, líquido e outras ondas de rádio	Devido o alcance curto, praticamente nenhuma
Automação	Geralmente necessita alguém para operar o *scanner*	Não precisa de intervenção humana	Talvez precise de intervenção devido à questões de segurança
Custo	Muito baixo	Baixo até alto (depende do tipo de *tag*)	Relativamente baixo

Apesar de o código de barras ser a solução mais barata e um leitor poder ser colocado dentro do armário de medicamento, não há flexibilidade para armazenar grande quantidade de dados e geralmente uma pessoa precisa operar o leitor. Outro problema é que somente um código pode ser lido por vez. Por outro lado, embora RFID e NFC sejam mais caros, elas possuem recursos que podem ser mais bem utilizados em um armário de medicamentos. Devido ao curto alcance das leituras NFC, a sua utilização não seria viável para o usuário, uma vez que seria necessário manter os objetos dentro desse limite. O RFID não é tão seguro assim, mas tem seus mecanismos para proteger os dados, como a criptografia. As suas duas

vantagens principais (essenciais para que seja apropriado para um armário de medicamentos) são a distância de leitura e a quantidade de itens que podem ser lidos por vez. O uso de *tags* passivas com um leitor com alcance de até 30 cm torna baixo seu custo. Além disso, várias *tags* podem ter suas informações extraídas ao mesmo tempo.

Dessa forma, a escolha da tecnologia RFID para atuar como uma interface para o usuário é uma solução vantajosa. Pois ela prove uma forma limpa e fácil para o usuário interagir com o sistema, ou seja, o usuário pode utilizar o armário de medicamentos como se fosse um armário de medicamentos comum, dado que as *tags* (prescrições, medicamentos e cartões de usuário) estejam dentro do alcance do leitor RFID.

Uma vez definida a interface de extração de dados do usuário, é necessário definir a interface para que o mesmo entenda as ações do sistema. Um sintetizador de voz é uma ferramenta que traduz texto escrito em língua falada e é aconselhável usá-lo em quatro situações: quando algumas tarefas exigem que o usuário esteja olhando para algo diferente de uma tela (e.g, caixa de remédio); quando há situações em que é necessário chamar a atenção (e.g., alarmes de medicação) do usuário; quando os usuários têm certas deficiências (e.g., deficiência visual devido à idade); e quando a aplicação tenta ter uma "personalidade", de modo que os usuários podem associá-la com características específicas (e.g., orientador de medicação). Estas situações, relacionadas com o processo de medicação, são a razão pela qual um sintetizador de voz foi escolhido para este trabalho de mestrado.

O processo de sintetização de voz passa por quatro estágios bem definidos que fazem parte de qualquer ferramenta que use tal tecnologia: análise estrutural do texto de entrada, pré-processamento de texto, conversão de texto para fonemas, análise de prosódia e, por fim, produção da forma de onda (áudio). Todavia, as APIs (*application programming interface*) das ferramentas são diferentes entre si, já que cada uma oferece recursos específicos. Dessa forma, para o armário de medicamento, a API SAPI (*Speech API*) da Microsoft foi escolhida como uma interface comum entre as diferentes ferramentas de sintetização de voz.

Para esta tese de mestrado, um sintetizador é necessário para instruir o usuário a tomar medicamentos. A solução na forma de um protótipo deve ter somente uma biblioteca que acesse a SAPI e não precise se preocupar com qual sintetizador foi usado. Contanto que o sistema operacional Windows seja usado, o código de aplicação do protótipo não tem de ser mudado caso um novo sintetizador seja usado, digamos, se houver a necessidade de se utilizar um sintetizador com uma voz de melhor qualidade.

Trabalhos relacionados

Nesta seção, nove trabalhos relacionados que abordaram o processo de medicação são descritos. Embora exista uma extensa lista de trabalhos nesta área, apenas aqueles que propuseram soluções para aumentar a adesão à medicação dos pacientes foram selecionados. Além disso, a pesquisa foi realizada com enfoque em oito fatores que têm impacto sobre a adesão à medicação, os quais são descritos a seguir: baixos níveis de conhecimento sobre saúde; fraco entendimento sobre o propósito da medicação; falta de importância dada ao impacto da medicação na saúde; falta de memória ou qualquer comprometimento cognitivo; falta de uma pessoa para acompanhar o tratamento do paciente; complexidade da medicação; comunicação fraca entre paciente e médico; pouca ou nenhuma condição de pagar por uma medicação. Como na versão em Inglês os trabalhos relacionados são descritos com detalhes, esta seção apresenta somente uma visão geral da comparação entre os mesmos.

Foi possível identificar dois grandes grupos de características presentes nos trabalhos relacionados: um grupo relacionado mais às necessidades tecnológicas de um sistema para auxiliar na medicação e um segundo grupo relacionado mais as necessidades do paciente em si. No primeiro grupo as seguintes características são consideradas: se o sistema pode identificar o usuário, medicamentos e prescrições de forma automática sem a intervenção explícita do usuário, assim como identificá-los de forma única (um elemento não pode ser usado para identificar outro); possibilidade de adicionar/remover vários itens de uma única vez no domínio do sistema, ou seja, capacidade de gerenciar múltiplas atividades das entidades ao mesmo tempo; detecção da ingestão de medicamento sem a necessidade de consultar o usuário; resposta rápida do sistema na computação do relacionamento das entidades para evitar que um remédio seja tomado de forma errada; possibilidade de o sistema ter pelo menos algumas de suas funcionalidades presentes em vários tipos de dispositivos (interoperabilidade); classificação clara de notificações (mensagens) em grupos distintos para o fácil entendimento do usuário.

Ainda no primeiro grupo, pode-se destacar: capacidade de fala com o usuário através de instruções bem claras e direcionadas; gerenciamento de medicamentos com *recall* ou que tenham sido anunciados como prejudiciais a saúde; recuperação do status de medicação, ou seja, mesmo que haja falta de energia elétrica, a medicação não é descontinuada (bastante importante no Brasil); uso do sistema por vários usuários ao mesmo tempo; interação natural com o sistema de forma que o processo de medicação seja o mais natural possível, sem que

haja a necessidade do usuário mudar muito seu cotidiano para tal; fácil entendimento, por parte do usuário, de como o sistema pode ajudá-lo; atualização de dados referentes aos medicamentos; e disponibilização do histórico de medicação através da internet (e.g, atrasos na medicação).

Já no segundo grupo, há as seguintes características: preocupação com a privacidade do usuário; controle para que crianças só tomem medicamento mediante a presença de um adulto; uso do sistema permitido somente para pessoas credenciadas; gerenciamento de incompatibilidades do usuário com medicamentos e entre remédios; gerenciamento do status do medicamento, se ele está vencido e a quantidade de pílulas restantes; alarme eficiente de medicação, de tal forma que o usuário seja lembrado continuamente e haja um plano de contingência (de acordo com a prescrição) para compensar os possíveis atrasos; abstração da complexidade da prescrição para que o usuário seja auxiliado e tenha que prestar atenção somente aos alarmes; flexibilidade para determinar o início da medicação e, em caso de esquecimento, uma janela de tempo para que o remédio seja tomado sem prejudicar a medicação.

Ainda no segundo grupo, é importante destacar: gerenciamento da dosagem do medicamento para que os limites diários de dosagem e quantidade de pílulas de acordo com a faixa etária sejam respeitados; engajamento do usuário, no qual o paciente aprecia o uso do sistema para tomar a medicação; a capacidade do sistema em trocar de idioma de acordo com a nacionalidade do usuário; para o caso em que uma pessoa que vive sozinha ou é idosa, a disponibilização de um canal de comunicação entre o paciente e médico (ou cuidador) é muito importante para o tratamento; e, por último, gerenciamento de reações medicamentosas com base no histórico de medicação do usuário (e.g. incompatibilidade com um medicamento previamente tomado cujo efeito ainda não terminou).

Como essa área que envolve a medicação de pacientes não é facilmente descrita de forma quantitativa, já que o processo de medicação está intrinsicamente relacionado ao usuário (i.e. suscetível a aspectos subjetivos), foi necessário reunir uma quantidade extensa de características qualitativas para serem abordadas na tentativa de melhorar a adesão à medicação em diversas frentes de necessidades do usuário. Já que, ao se analisar quais características foram contempladas pelos trabalhos relacionados, verificou-se que nenhum deles tratou, pelo menos de forma básica, todas elas.

A proposta do armário de medicamentos inteligente

Para visualizar melhor o conceito do armário de medicamento inteligente, primeiro é necessário entender a situação de referência do processo de medicação. Nos trabalhos relacionados foi possível verificar a existência de alguns elementos importantes envolvidos em tal processo: grupos de pacientes idosos e jovens, o uso de caixas de medicamentos ou garrafas e horários de medicação na forma de alarmes. Todos esses elementos, no entanto, só precisam ser representados por três entidades, que são o usuário, medicamento e prescrição ilustrados na Figura 1. Além disso, alguns bulários eletrônicos e médicos foram consultados para se fazer a análise abaixo.

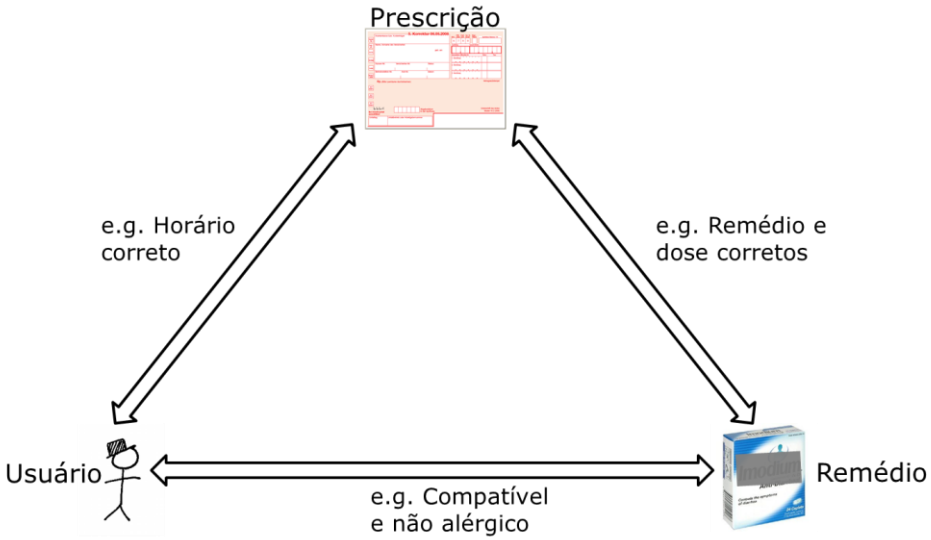

Figura 1: Situação de referência do processo de medicação

Ao analisar as entidades de forma separada, não é possível entender como o processo de medicação pode se tornar complexo para o usuário. Dessa forma, foi preciso analisar as relações entre as entidades. Entre o usuário e a entidade de medicamento, é importante considerar alergias, incompatibilidades com um ingrediente ativo ou grupo de ingrediente ativo do medicamento, e se a caixa não está vencida, vazia ou está sob o processo de um *recall*. Além disso, os dados de medicamentos tomados recentemente são importantes para evitar misturar com outros incompatíveis, caso o efeito do primeiro ainda esteja presente no organismo do usuário e garantir que a dosagem diária permitida seja obedecida, evitando overdoses.

Quanto às relações entre usuário e prescrição (que também envolve a relação da prescrição com o medicamento) alguns aspectos são essenciais, tais como a frequência em que o remédio é tomado (e.g., a cada oito horas), a dose certa para cada horário (e.g., dois comprimidos por ingestão) e a duração total do medicamento (e.g., tomar um medicamento por uma semana). Por fim, para o usuário, o horário certo para tomar o medicamento é importante.

Um dos problemas típicos que podem surgir com os horários da medicação ocorre quando o usuário pode esquecer o horário certo para se tomar um medicamento. O que deve ser feito sobre tal atraso não é uma solução trivial. Com as fontes mencionadas acima, foi possível identificar os seguintes cenários gerais para quando um horário é perdido: No primeiro, o horário perdido deve ser ignorado, seja continuando nos próximos horários regulares (adicionando-o para o fim da medicação) ou dobrando a dosagem do horário seguinte. Ou, no segundo cenário, o horário esquecido deve ser tomado imediatamente, seja reprogramando os horários restantes (com base no tempo do recém-tomado) ou continuando a medicação regularmente.

Como pode ser visto, há muitos parâmetros a serem considerados ao se tomar um remédio. Na maioria das vezes, o usuário não sabe toda esta informação técnica ou simplesmente a ignora. Por outro lado, com o sistema proposto neste trabalho, a pessoa não tem que lidar sozinha com todos estes parâmetros, já que há uma unidade para gerenciá-los. A Figura 2 abaixo mostra o conceito proposto.

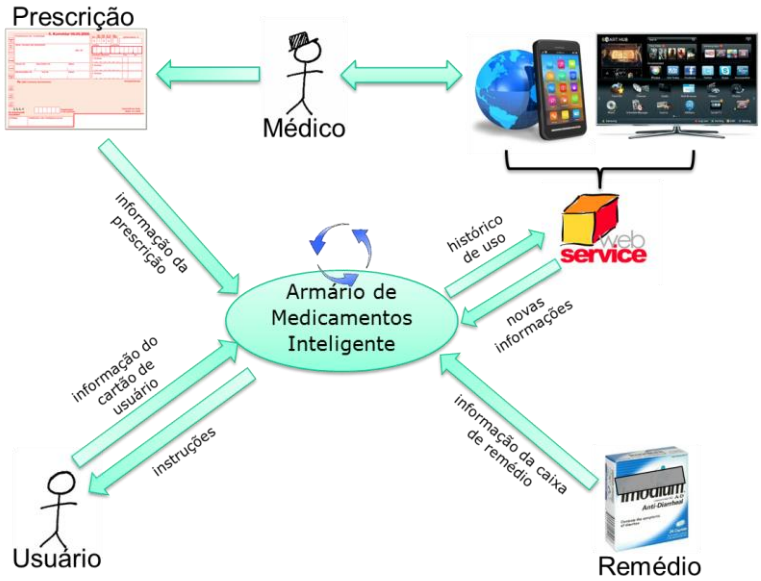

Figura 2: A proposta do armário de medicamento inteligente

A inteligência do sistema começa com a detecção automática das informações dos usuários, medicamentos e prescrições assim que eles estiverem ao alcance de leitura. Com base nas relações dessas entidades, o sistema proposto pode ajudar os pacientes a tomar o remédio certo e na hora certa, instruir e gerar lembretes (alarmes). Além disso, ele registra quais medicamentos são tomados, por quem, com que frequência e o horário tomado; disponibilizando essa informação através de um *web service* e permitindo que o médico monitore o progresso do paciente.

Os usuários, por sua vez, possuem cartões com suas informações. No verso das prescrições existem *tags* com detalhes sobre o tratamento (medicação) e no interior das caixas de remédios há *tags* com informações específicas de cada uma delas. Essas informações das entidades são detectadas usando uma tecnologia "sem fio" e devem ser previamente salvas para as *tags* (as prescrições devem ser armazenadas por um médico). O sistema gerencia dois tipos de medicamentos, aqueles que precisam e aqueles que não precisam de receita médica para serem tomados. Além disso, ele classifica prescrições que podem ser repetidas ou não (alguns medicamentos precisam de uma pausa entre os ciclos de medicação).

Uma vez que as informações das entidades estão disponíveis, o paciente pode usar o cartão para entrar no sistema, tornando-se um usuário ativo, que é um pré-requisito para que o armário seja usado. Se uma nova receita é adicionada, a sua informação é lida automaticamente e o paciente pode começar a tomar os medicamentos da sua lista. Quando um medicamento é adicionado ao armário, caso este esteja vencido ou tenha sido anunciado como perigoso ou esteja sob algum *recall*, o usuário é imediatamente informado sobre isso.

Para tomar um medicamento, a pessoa precisa apenas tirá-lo do armário e se há alguma razão pela qual ele não deve ser tomado, o armário informa sobre tal situação. Caso contrário, depois de alguns segundos, o remédio é considerado como tomado, podendo ser recolocado no armário e o usuário é informado sobre a quantidade de pílulas restante caso haja poucas na caixa. Se o medicamento precisa de uma receita médica para ser tomado, a informação de sua medicação é atualizada no sistema (horário da ingestão, se houve qualquer atraso e o próximo horário). No caso em que um medicamento não deve ser tomado, quando ele é retornado para o armário, não é considerado como tomado (já que o paciente foi advertido sobre o perigo). É importante ressaltar que o sistema atribui prioridade para os remédios com prescrição médica. Portanto, se o usuário tentar tomar um medicamento (que não precisa de prescrição) que é incompatível com alguma prescrição em uso, tal ação é rejeitada.

Além disso, o *web service* desempenha uma função de gerenciador de *recall*, no caso de um remédio ser anunciado como perigoso ou com algum erro de fabricação. Ele também fornece estatísticas do histórico de uso do armário de medicamento, permitindo que o tratamento dos usuários seja monitorado por um médico. Também é possível inserir novas informações sobre medicamentos para o banco de dados através deste serviço; por exemplo, informações de novas interações medicamentosas.

O armário também fornece flexibilidade, permitindo ao usuário escolher quando começar a medicação, lembrando-o dos horários a serem tomadas e gerando alarmes (lembretes) por um período pré-definido de tempo, até que o medicamento seja tomado. O usuário também é informado sobre o que deve ser feito no caso de o horário certo para tomar um remédio ser esquecido. Por fim, a proposta desse armário inclui ainda a administração da dose diária recomendada de medicamento, permite que uma prescrição possa ter mais de um medicamento em sua lista e pode administrar mais do que uma prescrição ao mesmo tempo.

Desenvolvimento do Sistema

Como mostrado anteriormente, o armário de medicamento inteligente pode ser utilizado pelo paciente e um médico. Para simplificar (e flexibilização) seus casos de uso são descritos a seguir contendo somente um usuário geral. A maioria dos casos de uso são destinados ao paciente, mas alguns deles são destinados principalmente ao médico. Reunindo as formas possíveis que esse ator pode interagir com o sistema, alguns casos de uso foram criados e são descritos abaixo.

Como parte do caso de uso "*colocar prescrição*", assim que uma prescrição é adicionada pelo usuário ao campo de leitura do armário, checa-se se é uma nova receita ou se já estava presente no sistema anteriormente. Se ela é nova, os dados são salvos e o usuário é informado que um medicamento pode ser tomado, caso contrário, ou a prescrição é considerada como removida do sistema se passou muito tempo ausente, ou sua medicação é reiniciada/finalizada dependendo se a prescrição pode ou não ser repetida. O caso de uso "*remover prescrição*" refere-se à retirada de prescrição do armário. Se for por um curto período de tempo, ela é considerada como retirada, mas se for muito tempo, é considerada como inativa.

Para o caso de uso “*colocar medicamento*”, quando um remédio entra no alcance de leitura do armário, ele é considerado presente no sistema e o usuário é informado de três possíveis situações: que o medicamento foi tomado, que um novo medicamento foi adicionado ao sistema ou que o remédio não foi tomado. O primeiro caso ocorre quando a remédio tinha sido retirado por um tempo suficiente para ser tomado e é estendido pelo caso de uso "*tomar remédio*". O segundo caso acontece quando é verificado que o medicamento não havia sido adicionado anteriormente e é estendido pelo caso de uso "*adicionar novo medicamento*". E o terceiro caso acontece quando houve tempo suficiente para se tomar alguma pílula, mas verificou-se que havia alguma incompatibilidade pela qual seria prejudicial para o usuário tomar o medicamento.

Para o caso "*remover medicamento*", depois que um medicamento deixa o alcance de leitura do armário, três situações podem acontecer com base no tempo decorrido até que o medicamento seja colocado de volta: se permanecer removido por um curto período de tempo (não suficiente para tomar qualquer pílula) nada acontece; se permanecer fora do armário por um período de tempo suficiente para se tomar o medicamento, o caso de uso "*tomar remédio*" é usado; e se permanecer afastado do armário por um longo período de tempo, o caso de uso "*remover medicamento do sistema*" é utilizado.

O caso de uso "*tomar remédio*" acontece depois que o medicamento foi retirado por um período de tempo suficiente para considerá-lo como tomado, então o sistema verifica se o usuário pode tomá-lo e, em caso positivo, é registrado no sistema que ela ele foi tomado, independentemente de ser um medicamento com ou sem receita médica. Se for com receita médica, as informações relacionadas, como o próximo horário, também são atualizadas. Por outro lado, se for verificada qualquer incompatibilidade do medicamento com o usuário, ele é informado disso juntamente com a razão. Este caso de uso estende ambos os casos de uso "*colocar medicamento*" e "*remover medicamento*", uma vez que depende de ambos para acontecer.

“*remover medicamento do sistema*” acontece quando o remédio sai do alcance de leitura do armário por um período de tempo maior do que o necessário para tomá-lo, tendo suas informações removidas da memória do sistema. Já o caso de uso “*fazer login*” é um pré-requisito para que o armário seja usado, sendo necessário que um cartão seja colocado ao alcance de leitura. Quando o caso de uso “*abrir a porta do armário*” é usado, o sistema verifica se alguém efetuou *login*. Uma mensagem de alerta é gerada ao se abrir a porta caso ninguém tenha feito *login*, ou caso uma criança tente usar o armário sem ter outro usuário

(adulto) com *login* efetuado naquele instante. Se não houver qualquer problema, o usuário recebe uma mensagem de boas vindas juntamente com sinais visuais.

O caso de uso "*tornar-se usuário ativo*", como o nome diz, faz com que o usuário se torne ativo após duas situações: ele foi o último a efetuar *login* no sistema ou havia logado exatamente antes de um usuário que faz *logout* no momento. É possível ver que o sistema enfileira os *logins* em um estilo LIFO (último a entrar, primeiro a sair). O usuário ativo é importante para que o sistema detecte quem está usando-o no momento, caso contrário, se não houvesse tal delimitação, seria impossível saber quem está interagindo com o sistema e ambiguidades poderiam surgir. O caso de uso "*log out*" acontece quando o cartão de usuário deixa o campo de leitura, o usuário é desconectado do sistema e o caso de uso "*atualizar usuário ativo*" entra em ação. Quanto ao caso "*atualizar usuário ativo*", quando um usuário efetua *logout*, é verificado quem era o outro usuário conectado anterior a ele, se houver, e então este último se torna o usuário ativo.

Como operações extras, o usuário pode "*requisitar informação*" do armário de medicamentos. Tal requisição é feita através de um *web service* e pode ser usada para monitorar o progresso da medicação do paciente. Esse caso de uso é estendido por "*requisitar estatísticas*" e "*requisitar status de operação*". O primeiro refere-se ao acesso ao histórico de uso do paciente, já o segundo refere-se ao acesso ao estado de funcionamento do armário (e.g., se ele parou de funcionar por algum motivo). Além disso, o usuário pode "*adicionar nova informação*" ao sistema, caso de uso estendido por "*adicionar informação às tags*" e "*adicionar informação ao banco de dados*". O primeiro permite inserir informação ao cartão de usuário, receitas e caixas de remédio. Já o segundo permite inserir informações mais gerais sobre os medicamentos (e.g. nova interação medicamentosa).

A Figura 3 demonstra a arquitetura do sistema feita com base nos casos de uso. Ela pode ser dividida em quatro partes principais: abstração de dados das entidades, que é composta pelas entidades usuário, prescrição, medicamento e da base de dados do sistema; leitura de RFID, que é composta pelo leitor RFID e sua antena; notificações, por sua vez composta pela placa do microcontrolador, sensor da porta do armário, saída de voz e sinais visuais (LEDs); *web serice*, que é composto pelo banco de dados do sistema, script PHP e JSON, que fornecem uma interface para acessar as informações do armário de medicamentos via web; e lógica de programa escrito em linguagem de programação Java, cuja função é fornecer uma medicação suave e sob orientação para o usuário.

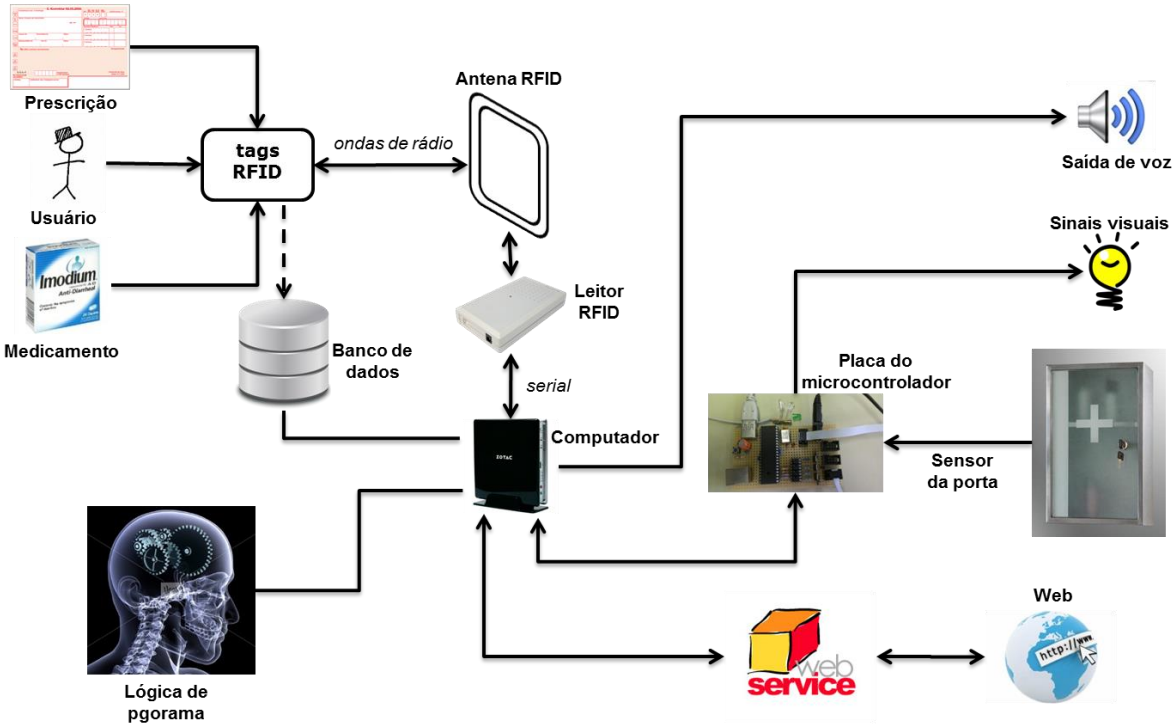

Figura 3: Arquitetura do armário de medicamento inteligente

Em linhas gerais, a arquitetura segue o seguinte fluxo. No canto superior esquerdo as entidades têm suas informações, armazenadas nas *tags*, enviadas por ondas de rádio para a antena que está conectada ao leitor de RFID. Uma vez que tal informação é extraída das *tags*, ela também é armazenada no banco de dados do sistema. O computador, que está conectado a todos os elementos da arquitetura, desempenha o papel central de processar toda a informação proveniente das entidades e enviar sinais, via comunicação USB, para a placa do microcontrolador (LEDs e sensor de contato). O *web service*, por sua vez, lê informações do banco de dados e as disponibiliza para a web.

Protótipo

A Figura 4 mostra o protótipo do armário de medicamento inteligente. Na frente dele, apenas os LEDs são visíveis para fornecer sinais visuais para o usuário. O hardware restante fica na parte de trás, de modo que o usuário não precisa se preocupar em como ele foi feito. Tudo que o usuário tem que fazer é usar o seu cartão e começar a usá-lo como um armário de medicamento convencional, já que todo o gerenciamento é feito automaticamente. Ele pode,

contudo, ser fechado com uma chave, de modo que o gerenciamento de quem o usa possa ainda ser feito manualmente caso falte energia elétrica.

Figura 4: Protótipo do armário de medicamento inteligente

Em termos gerais, o armário de medicamento inteligente pode ser dividido em duas partes principais, que são representados na Figura 5: domínio do sistema e do usuário. A primeira refere-se aos elementos abstratos ou físicos que podem ser acessados diretamente pelo usuário. Dentre tais elementos há a caixa de remédio, o cartão de usuário, prescrição, porta do armário, luz indicadora, instruções (via voz) e qualquer dispositivo/software que acesse o *web service*. Já a segunda parte é composta de elementos de hardware/software que permitem que o armário gerencie os relacionamentos das entidades e interações do usuário.

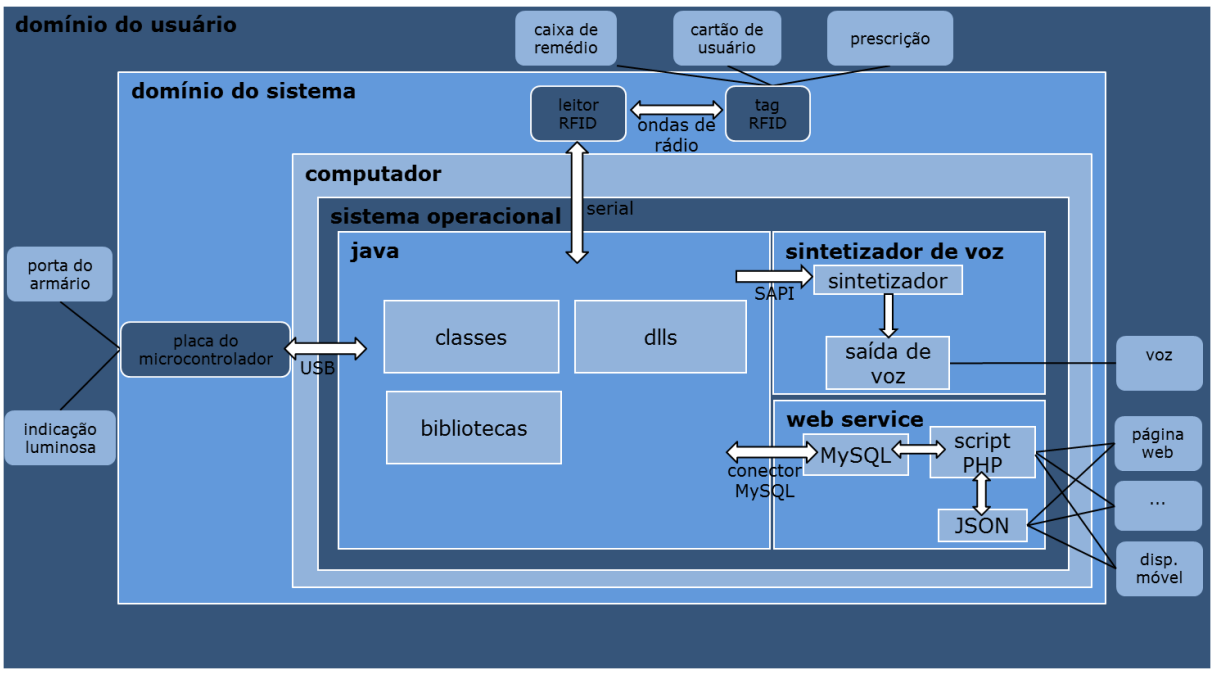

Figura 5: Visão geral do protótipo

Um computador é usado para executar o código-fonte, o *web service* e o banco de dados. As entidades são detectadas sem fio usando a tecnologia RFID e, para armazenar suas informações, *tags* RFID passivas do tipo ISO15693, com uma frequência de 13.56 MHz, foram utilizados. Este tipo de *tag* também pode ser lido através de aplicações NFC em um dispositivo móvel.

Para detectar as *tags*, um comando é enviado pelo computador aproximadamente a cada 250 ms para que o leitor RFID obtenha as IDs das *tags* próximas. Se uma *tag* é adicionada ao alcance de leitura, outro comando é enviado para obter os seus dados, completando um período de cerca de 300 ms para todo o processo, caso contrário, apenas as IDs são lidas. Se uma *tag* for removida, os dados capturados quando ela foi adicionada são usados. Para saber quais *tags* foram adicionados ou removidos, o sistema mantém uma lista temporária das *tags* ao alcance de leitura, de modo que, na leitura seguinte, há uma lista para se fazer comparações. Assim, o sistema detecta quando prescrições, cartões e medicamentos dos usuários são adicionados ou removidos do armário e toma uma decisão em cima disso.

Através da combinação de informações das entidades, o sistema instrui o usuário (por exemplo, se um medicamento vencido é retirado, o usuário é avisado sobre isso). A lógica do programa também compara as novas informações com um banco de dados MySQL, dessa forma, como um exemplo, é possível saber se o usuário tomou anteriormente um medicamento incompatível com outro que está sendo retirado e um alerta é gerado . A linguagem de programação Java foi usado para implementar a lógica principal do programa e, usando bibliotecas auxiliares, foi possível acessar APIs relacionadas com o sintetizador de voz, comunicação serial e o conector do banco de dados.

O programa principal tem uma classe exclusiva para acessar o banco de dados do sistema usando um conector MySQL. O sintetizador pode ser acessado através da SAPI. Como podeser visto na Figura 5, o sistema foi modularizado de tal maneira que o sintetizador de voz pode ser substituído por outro desde que tenha uma interface com a SAPI. Um microcontrolador da família PIC foi integrado em uma placa, para obter o sinal de um sensor de contato (que detecta se a porta está aberta/fechada), com o qual o sistema combina informações do cartão de usuário para concedê-lo acesso, e para ligar/desligar LEDs (vermelho, verde e branco) que indicam, de forma categorizada, quando o usuário deve prestar mais atenção ou não deve tomar um medicamento, por exemplo. A linguagem C foi utilizada para programar o microcontrolador.

Além disso, há o *web service* composto da base de dados e um módulo PHP/JSON. O banco de dados é a interface comum entre o programa principal e tal serviço, porque é onde todas as informações de uso do armário estão concentradas. Na Figura 5 pode-se ver que o serviço pode ser usado por uma página web, por dispositivos móveis e por qualquer outro dispositivo/software que possa enviar solicitações HTTP para ele e ler respostas em JSON.

Como exemplo de um cenário possível, se alguém, sem um cartão de usuário válido abre a porta do armário, esta ação é detectada pelo sensor de contato e um sinal é enviado para o computador através do microcontrolador. O computador, em seguida, sintetiza uma mensagem de voz e envia um comando para o microcontrolador para piscar LEDs vermelhos, mostrando que essa ação não é permitida. O mesmo aconteceria se uma criança, mesmo com um cartão válido, tentasse usá-lo sem um adulto conectado ao mesmo tempo. Por outro lado, se o usuário tem um cartão válido, o protótipo o detecta no alcance de leitura e pisca LEDs verdes para mostrar que o acesso foi concedido. Assim que o usuário abre o armário, esta ação também é detectada e uma mensagem de boas-vindas é gerada junto com sinais dos LEDs verdes.

Resultados e conclusão

Para determinar se esta dissertação de mestrado alcançou resultados positivos e que realizou o que foi proposto no início deste documento, os seguintes tópicos devem ser levados em consideração: as características que foram levantadas nos trabalhos relacionados, já que elas resumem o que foi proposto pelo estado da arte; os objetivos gerais e específicos desta tese; e os requisitos do protótipo. Ao fim do desenvolvimento do sistema, embora alguns poucos tópicos tenham um limite de aplicação (e.g., detecção automática da ingestão de pílulas), todos os objetivos e requisitos foram comtemplados com êxito.

Os trabalhos relacionados discutidos nesta dissertação de mestrado mostraram como o processo de medicação pode tornar-se perigoso para o usuário, devido a seus muitos parâmetros e medicações complexas, e propôs o conceito de um armário de medicamento inteligente que usa a tecnologia RFID para identificar automaticamente as informações do usuário, de medicamento e de prescrição. Ao processar as relações entre estas entidades, ele gerencia as alergias do usuário, interações medicamentosas, data de vencimento de remédios e quantidade restante de pílulas.

A vantagem de usar esse conceito é que os usuários não têm que cuidar sozinhos dos diversos parâmetros envolvidos no processo de medicação, já que são instruídos pelo armário através de um sintetizador de voz e sinais visuais. Todas estas características são uma tentativa para resolver o problema da falta de adesão à medicação. As instruções são usadas para dizer se um medicamento deve ou não ser tomado, o momento certo para tomá-lo e o que deve ser feito caso um horário seja perdido. Além disso, ele gerencia a quantidade diária máxima de medicamento permitida e somente usuários válidos podem usar o armário, evitando o uso indevido por aqueles que não têm um cartão para serem identificados ou por crianças não supervisionadas por um adulto.

Outra grande funcionalidade que foi desenvolvida para o armário é a sua capacidade de trabalhar *offline*, pelo menos parcialmente. Através da utilização de uma fechadura manual, os pacientes não são impedidos de utilizar o armário frente à falta de energia elétrica. O mesmo princípio se aplica ao gerenciamento das prescrições, já que, apesar de os alarmes não serem gerados por falta de energia, o sistema tenta recuperar o status da medicação enquanto o armário estava desligado. Este modo de recuperação considera que durante a falta de energia, o usuário, interessado em sua saúde, assumiu o gerenciamento manual de sua medicação. Dessa forma, quando o sistema é ligado, ele considera que o medicamento foi tomado normalmente. Este recurso é muito importante no Brasil, especialmente em cidades onde a falta de energia é um problema constante.

Ao longo do desenvolvimento do armário de remédios inteligentes muitos desafios foram enfrentados. O primeiro foi encontrar um sintetizador de voz profissional com baixo custo. A maioria dos mecanismos de fala custa cerca de 1000 € e alguns deles só fornecem licenças temporárias. Depois de uma série de negociação e testes de APIs, verificou-se que a solução mais barata era comprar um software que usa a maioria das vozes disponíveis no mercado e integrar as vozes programaticamente. Para alcançar este objetivo, as vozes necessárias precisam ser compatíveis com a API SAPI da Microsoft.

Outro desafio, talvez o mais difícil, foi o leitor de RFID, já que o sistema tinha que ser concebido de tal forma que fosse possível ter um processo dedicado para ler as *tags* RFID. Com isso, vários tratamentos foram inseridos para lidar com os erros induzidos não apenas por *tags* mal posicionadas ou em uma região de leitura instável, mas também pelas ações do usuário, uma vez que o armário deve ser usado de forma convencional. Outro problema é o modo como o Windows permite a comunicação serial: como existem muitas *threads* em execução no sistema operacional, não há um comportamento determinístico para se ler os

dados. Dessa forma, muitos testes foram feitos para determinar a frequência correta em que os dados devem ser solicitados a partir do leitor de RFID. Além disso, muitas possibilidades de interação foram testadas para obter informações úteis sobre a interação do usuário.

O terceiro maior desafio foi o desenvolvimento do conceito de prescrição. Muita pesquisa e leitura (incluindo consultas a médicos) foram necessárias para avaliar quais são as ações mais comuns a serem executadas quando o usuário esquece de tomar um medicamento na hora certa. O comportamento temporal das prescrições aumentou o nível de dificuldade tanto no cálculo da programação quanto na reconstrução da medicação do usuário (em caso de falta de energia).

O último desafio refere-se ao fato de que, como esta tese de mestrado foi feito em parceria com o "Institut für Automatisierungs- und Softwaretechnik", o modelo de engenharia de software do instituto (modelo-V) foi usado para manter a rastreabilidade do processo ao longo de todo o mestrado. Entre as dificuldades enfrentadas lá, pode-se destacar: o idioma, já que para a confecção da placa do microcontrolador e instalação do sistema operacional o idioma Alemão teve que ser utilizado; e o instituto representou um "cliente" muito exigente, já que o armário é parte de seu *showcase* (i.e., é de interesse público), o que significa que houve bastante pressão para entregar algo funcionando corretamente.

Como trabalho futuro, mais idiomas podem ser incluídos (e.g., espanhol). Outras tecnologias podem ser utilizadas para identificar o usuário ao invés de usar apenas um cartão de RFID. Uma câmara poderia ser usada para reconhecer o paciente ou a ingestão do medicamento de forma mais precisa. Com o *web service*, informações dos medicamentos provenientes da web poderiam ser inseridas no banco de dados automaticamente. O armário de medicamento poderia ter tal serviço estendido para permitir atualização do código fonte do sistema ou verificação mais detalhada do status de funcionamento do armário.

Outro passo importante como trabalho futuro é testar o armário com usuários reais de diferentes faixas etárias. Com seus *feedbacks*, erros podem ser encontrados e certas características podem ser melhoradas (as desnecessárias removidas e as complexas simplificadas). Com usuários reais, pode-se ainda verificar a real eficácia do sistema, ou seja, se ele de fato reduz a falta de adesão à medicação.

Por fim, o armário de medicamento inteligente é um conceito desenvolvido para facilitar a vida das pessoas ao se tomar medicamentos, de tal forma que ele possa ser utilizado de forma tão simples quanto usar um armário convencional, todavia aumentando a adesão a medicação. Ao usá-lo, se um paciente teve que aprender quase nada para tal, o objetivo

principal desta tese foi alcançado: ser o mais próximo possível do que as pessoas esperam de armário de medicamento ao se tomar remédios, proporcionando um processo de medicação suave, fácil e sob orientação.

Chapter 1- Introduction

As the world has seen improvements on different areas of the healthcare, increasing global life expectation and more availability of powerful drugs on the market to treat a variety of health problems, it has also seen an increasing number of related problems. People started to take more drugs by themselves, without even consulting a doctor, increasing the occurrence of drugs side effects and drug-induced deaths.

Some newspapers have reported on this problem, which may be evidenced by the following news: "*Parents sue after toddler son 'pukes up blood and dies after taking recalled Children's medicine*" [1]; "*A Colorado girl is dead after taking a lethal combination of two common cold and allergy medicines*" [2]; "*multiple doctors can prescribe multiple drugs with little awareness of what anyone else has prescribed*" [3]; "*A grandmother suffering from chronic back pain apparently forgot she'd already taken her daily regimen of pills and ended up double dosing*" [4].

Drug-induced deaths are something more common than it could be imagined and they can be evidenced by: "*In some ways, prescription drugs are more dangerous than illicit ones because users don't have their guard up*" and "*Drug deaths now outnumber traffic fatalities in U.S.*" [4][5]. Thus, a common and simple activity as taking medicine, without proper guidance, may be harmful to people and even mean risk of death.

It is clear there is a need for healthcare solutions to improve medication adherence. Their conception, however, is not trivial when usability for the final user matters. The challenge in developing an accessible and usable system arises from the fact that capturing all user's properties, limitations and wishes is a hard task due to the different users' expectations on a system or product. An area that deals directly with this is Ambient Assisted Living (AAL). It aims to assist the elderly people to live independently, extending the time they can live in their home with autonomy and having activities of daily life carried out by intelligent assistive devices [6]. Different points of view and levels of cognitive understanding are among the challenges to implement such environment at home.

However, by using "home-based" healthcare solutions, on which the availability of biomedical devices from the hospital would be shifted to the user's home, accessibility could be improved [7]. Compact devices could be deployed to improve patient's outcome and

potentially control healthcare costs by allowing screening and monitoring in home setting. Nonetheless, healthcare monitoring at home is much more complex because of specific home constraints and people's habits [8].

In the late 60's and early 70's, the main obstacles faced by these healthcare solutions were related to hardware limitations; but nowadays, due to advances in technology, the obstacles are different [9]. One example is the lack of interface between the systems and the information provider, in which a system is used only by the patient and a nurse, negating the possibility of providing medical guidance to the doctor, such as warning about the dangers of combining specific drugs presented in different medications [10].

In [11], three main characteristics are suggested as good for healthcare interfaces: customization can be done according to different people; better information leads to better treatment, so that the information is present when the decision-making is needed; as the user is not always skilled at computers, the interface should provide different templates.

These suggestions are an attempt to increase usability and accessibility. The ISO/TS 16071 distinguishes them as follows: accessibility alone refers to providing device for everyone, while usability first defines the target audience and, based on that, tries to provide accessibility to all of them [12].

Back to the medication problem, taking medicine is an activity that is very common in an AAL context and is the target problem of this thesis. In general, elderly people take more medication and are more prone to forget to take it at the right time or have difficulty in reading medical information about medicines than a younger person. But those are not the only reasons for which medication adherence may become a problem of great importance, as younger people have also faced problems in understanding the medical background involved on this process.

Nevertheless, there is still missing consolidated solutions aiming the monitoring and control of medication adherence. On this master thesis a healthcare solution to assist people in taking medicines is going to be formulated and described in details. Even though computerization of healthcare applications offers the best and easiest approach to provide medical guidance and allow appropriate data capture [10], the way it is provided to the patient must be well structured. Thus, the intelligent medicine cabinet prototype proposed on this work was designed to be as simple as using a common medicine cabinet, increasing its usability in an AAL context.

1.1 Motivation

To support the user in taking medicines, one could think of a simple alarm device, through which the user is alerted on the right time to take a medicine. But such type of management would still have problems, such as the need of programming the alert device on every medication or even the lack of control if the patient really took the medicine, much less if the right medicine is taken. Another shortcoming would be the lack of an interface, through which the doctor, or person in charge, could access such information. Nowadays the doctor usually has only the information provided by the patient, in other words, if it is wrong, the doctor may evaluate the patient's treatment in a wrong way.

Another very common shortcoming in the area is related to the prescription. In Brazil, for example, the only way through which the user may know which medicines that should be taken (as well as the right time and dosage) is via a prescription which is usually handwritten or on a printed paper. The problem with the handwritten version is that the user may not clearly understand the medical specifications and take an action harmful to their treatment. As for the printed version, the user may easily lose it, without having a way to recover the data. Actually, this is a common problem for both types of prescription.

To complete the scenario, the world is getting older and older. This fact was pointed out by [13], which revealed that the global population of the older people is growing at a very fast rate. As predicted by The United Nations Population Division [14], the population aged 60 or over will increase by more than 50% over the next four decades. An older population will need more home healthcare and, as the medication plays an important role on the elderly daily lives, solutions to overcome the problem of poor medication adherence will be very important on such context.

The problems mentioned above and in the introduction show that a natural assistance that aims on the maximum user acceptance would be an appropriate approach to increase medication adherence. Such requirement, however, brings many challenges in determining which kind of assistances (e.g. intake confirmation by user or automatic detection) should be provided. All these challenges serve as a motivation for the work developed on this master thesis.

1.2 Objectives

1.2.1 General Objectives

Investigate the conception of and realize a system in an Ambient Assisted Living context which implements an electronic healthcare solution through the development centered on the needs of user medication and on the realization of an intelligent medicine cabinet prototype.

1.2.2 Specific Objectives

1) Investigate and evaluate the strong and weak features of existing systems;
2) Propose a concept based on the state of the art;
3) Evaluate how existing healthcare system could improve the use of information collected from users, medicines and prescriptions;
4) Based on existing systems, define a solution with more use cases, trying to cover the image of the doctor in supporting and/or monitoring the patient medication;
5) Apply user centered automation techniques in an attempt to diminish the possibility of system failure, given the solution importance to the user health;
6) Define and develop an electronic healthcare solution with a friendly user interface to increase accessibility and usability for the medication process;
7) The proposed prototype should serve as a basis to apply the concepts of this master thesis aiming medication adherence.

1.2.3 Outline of this document

This master thesis is organized as follows. Chapter 2- shows the two technologies that are relevant to the development of the prototype of this master thesis: first the RFID technology is described, a comparison with other possible technologies with the same objective (i.e. automatic identification and data capture) is made, and the reason for which it was chosen is explained; later, it is shown how speech synthesizers work and how they can be

integrated into an operating system. Chapter 2- is important for the reader to understand the technological background necessary for the rest of this document.

In Chapter 3-, the state of art is investigated and the works related to the problem of medication non-adherence are described, showing how each one of them approached such problem. In the end, all of them are compared into two tables, showing which medication issues were covered or neglected by each one. These issues, based on the state of art, are exactly the ones which are explored by this master thesis.

In Chapter 4-, the medication baseline situation is described, showing into details the factors that may become complicated to the users and, consequently, that contribute to the medication non-adherence. Based on such factors and on the issues related on the state of art, the concept of an "intelligent medicine cabinet" is proposed, which is the subject of this master thesis. Thus, Chapter 4 is a key section to understand what the problem is, how it is linked to the issues of the related works, and how it should be solved by this thesis.

In Chapter 5-, the concept is used to design a system. It is shown the system use cases (i.e. how the user may interact with it), how it was designed, and its architecture. The latter, in turn, describes how the entities involved (i.e. user, prescription, and medicine), the RFID automatic identification, notifications to the user, the intelligent medicine cabinet, and its web service were modeled in an attempt to solve the non-adherence to medication by presenting a system that is intended to provide usability and accessibility.

Chapter 6-, however, shows how the implementation of the system was realized. It is first shown that it looks like a conventional medicine cabinet, but, on its back, the whole system performs a lot of functions to guarantee a smooth medication process to the user. These functions are later grouped into three categories (i.e. related only to the user, prescription, and medicine) and are described individually. Furthermore, an error correction section shows how the prototype deals with problems inserted by the user. Thereby, with chapter 6, a concrete solution is presented, so that this thesis does not cover only theoretical aspects, but has a practical appeal.

Chapter 7- refers to the issues discussed on the section of the state of art and to the specific objectives of this master thesis, showing how each one of them were in fact implemented. While most of them could be fully implemented; one or two needed an extra description about to what extent they were solved. Finally, on Chapter 8- an overview of what has been achieved with this master thesis is presented, the challenges faced during it (especially during its development phase), and some future works are listed.

Chapter 2- Relevant Technologies

In this chapter the technologies used on this work are described into more details. The first part shows how RFID technology can be used for automatic identification of objects and why passive tags were chosen to be used rather than active ones. A comparison is also made among RFID, barcodes and NFC technologies, showing what features of RFID technology were relevant to this master thesis, consequently, the reason why it was chosen.

The second part shows the speech synthesizer technology, describing how the speech outputs are formed. Later, a brief explanation is given on how the synthesizer may be used via a text to speech tool. In the end, it is shown how this tool may be integrated into operating system used by the solution proposed on the master thesis.

2.1 RFID and other wireless technologies

RFID is part of a class of technologies whose main purpose is automatic identification and data capture (AIDC). This technology is composed of two main components, through which it is possible to make the identification of objects or people: a reader and the tag. Among the many methods of identification, the most usual is to store on the tag its ID, which identifies the objects uniquely, but it is also possible to store data related to it [15]. A medicine tag, for example, would have stored on it an identification number and related data, such as the name of medicine, expiry data, quantity of pills and active substances.

By the use of radio waves, a remote communication may be established between the reader and the tag, also called RFID transponder. This transponder is composed of an antenna and a microchip (see Figure 1), being the antenna the way through which the transponder sends data and its ID to the reader. This information in radio waves is interpreted by the reader and converted into digital format, enabling the application to handle the information according to its use.

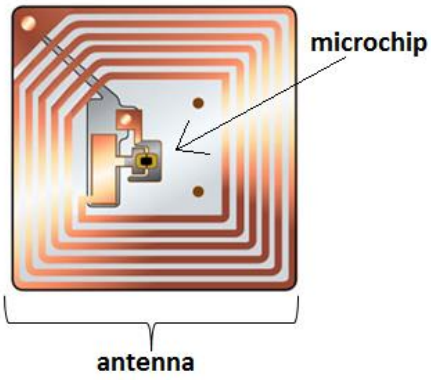

Figure 1: RFID transponder

The tags may be produced in many formats or encapsulated in different ways. That depends on the problem to be solved. For a user card it is better to have a plastic encapsulation in a card format than a paper encapsulation in a triangular format [16].

There are three types of transponders:

- Active – they do not need the radio waves from the reader to be energized, as they use an internal battery to such purpose. They use their own power source to run the microchip circuit and send signals to the reader. They are, thus, the most expensive ones and may achieve longer distances for transmissions.
- Semi-passive – they also have an internal power source to run the microchip circuit, but they use the radio waves energy from the reader to transmit their data. They are slightly less expensive and the distance for reading is a bit less than with the active ones.
- Passive – they are the most common type of transponders and, thus, the cheapest ones. They have no battery and depend completely on the radio waves that the reader sends to induct current on them to transmit data.

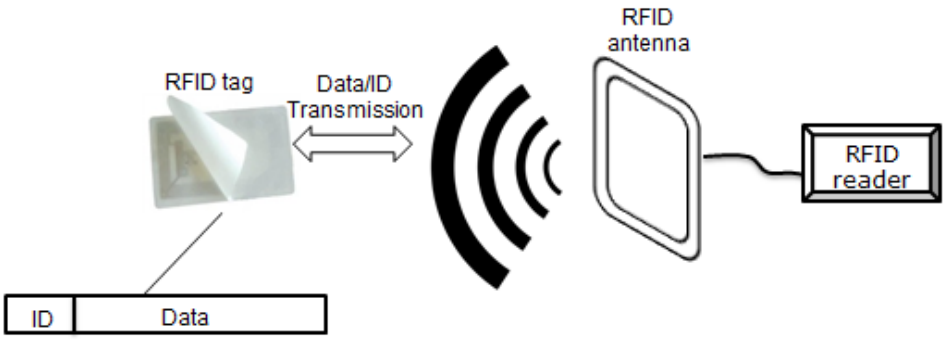

Figure 2: RFID Reading process

Figure 2 shows an example of transmission using a passive tag. The RFID reader uses its antenna to send radio waves into the environment. If there is any tag within the reading range, it gets energized using the radio waves that were sent and responds with its ID, also in the form of radio waves, to the antenna. The reader interprets the received signal and, with the information of the tag ID, it may send other radio waves with this ID being included to request the tag data. The tag gets energized again and, as it is being addressed with its ID, it sends its data back to the antenna [17]. As it can be seen, it is a very simple process to get information from the RFID tags.

In applications where tags need to be read from longer distances, the first two classes of transponders are necessary. They can be used to track cars information on a highway or train information in a railway. But, in applications like a medicine cabinet, the passive ones are enough, which also reduces its cost. Another technology which makes part of the AIDC group is the barcode. As the name suggests, barcodes are a pattern of black and white bars which encode information [18].

Even though there are several types of barcode schemas, Figure 3 shows one of the simplest to be understood. The first 6 digits of the figure represent the product/object manufacture code; the following 5 digits are the product/object identification; and the last digit is the verification digit, which is used for the barcode scanner to check if the reading of the code was right. The numbers depicted on the figure are actually the human readable version of what is encoded on the black and white bars.

Figure 3: Barcode example

Figure 3 is an example of a 1-D barcode, with which it is possible to understand how its encoding works. The first three bars, which represent the start of reading, are respectively black, white, and black. The reader uses them to know the standard spacing of the black and white barcodes. The other bars are the code itself formed of the combination of these three first bars. In the end, there is a stop bar, to know where the reading ends.

As mentioned before, there are other types of barcodes: some are numeric only (e.g. EAN – European Article Number); with a fixed length (e.g. EAN-13 with 13 digits); some can have numbers and alphabetic characters (e.g. Code 93, Code 128); and 2-D barcodes allows a lot of data to be encoded into smaller areas (e.g. QR code) [19]. Even though there are many types, they are all already standardized and their use is largely spread.

Depending on the type of barcode, the scanner (optical laser) gets electrical signals of the light reflected by white bars and absorbed by the black ones, therefore, one barcode must be read at a time. For the 1-D type, as an example, the scanner measures the relative widths of the bars and spaces (white bars) using the first three bars as a basis, translates the different patterns into regular characters and sends them to a computer or terminal. Here it is seen that a barcode does not store data about the objects, but only identifies them, as their information is actually stored on a computer or terminal. Another important thing to highlight is that the objects are not uniquely identified, as the identification number represents a class of objects/products.

A third technology in the AIDC group that could be used in a medicine cabinet is NFC, which stands for Near Field Communication. In short terms, NFC is a technology which allows wireless communication between two devices through the proximity between them, without the need of setting up a connection. This means, that as soon as the devices get in range, the connection is established automatically and data may be transferred.

Besides this technology may be used in a variety of applications, its strongest feature, for a technology which uses radio frequency, relies on security. The maximum connection distance of 10 centimeters is already one way to avoid exposing the transferred data. Some applications already use it on smartphones to make bank operations, such as paying a bill using the cellphone instead of a credit card. Other characteristic that allows more security is that, besides the 10 cm distant, the communication cannot be established in any angle, as the electromagnetic field is very short.

As NFC uses radio frequency for identification, it used the RFID standards in 2002, when it was created. But in 2003 it already had its own standard norm ISO/IEC 18092, which is still based on the ISO/IEC 18000 [17] and on the ISO/IEC 14443[20]. The first belongs to the RFID devices which operate at 13.56 MHz and the second defines the ID cards used to store information. NFC works using two types of ‘devices’: the *initiator* and the *target*. The first one is in charge of starting the communication and controlling the data transfer. The second must respond to the requests made by the *initiator*.

As NFC is part of the already consolidated RFID technology, it also has the passive and active transmissions mentioned above. For a medicine cabinet, the passive transmission would be more appropriate, as it would be possible to insert tags into packages and cards, which would be powered by the *initiator* radio waves and respond to its requests (readings). Besides of these two types of transmission, there are still three types of operation [21]:

- Reading/Writing – using the passive transmission, it allows the reading and modification of data in NFC devices;
- Peer-to-peer – it is a bidirectional communication used to exchange information between two devices. It allows that both devices either send or receive data from and to each other. This mode may be used to exchange files between devices;
- Card emulation – in this mode, a NFC device may work as a smart card in such a way that the reader may not distinguish between a smart card and a mobile device.

Table 1 shows a comparison among these three technologies. Taking a look at this table, it is possible to see the strongest feature of each technology. The barcodes are extremely cheap compared to the other two technologies; RFID allows the reading of many items at a time; and NFC provides different ways of securing data.

Even though the barcode is the cheapest solution and a scanner could be placed inside the medicine cabinet, it is not very flexible to store large quantity of data and usually needs a person to operate the scanner or the barcode under the laser to read it. Another problem is that only one barcode can be read at a time.

RFID and NFC technologies work in a similar way. Even though both are more expensive than barcodes, they contain some extra features which may be interesting for a medicine cabinet or not. Due to the short range of the NFC readings, up to 10 cm, its use would not be practicable for the user, as it would be necessary to keep the objects within this range. The RFID is not so security designed, but it has its mechanisms to protect data, like encrypting. Its two main advantages (crucial for it to be suitable for a medicine cabinet) are the reading range and the quantity of items that can be read at a time. Using passive tags with a reader reading up to 30 cm distant makes its cost be low. Besides, many tags can have their information extracted at the same time.

Table 1: Comparison among RFID, barcode and NFC technologies

	Barcode	RFID	NFC
Line of side	Required	Not required	Not required
Read range	From few centimeters up to some meters	Up to 100m with active tags	Up to 10cm
Read rate	One at a time	Up to many at a time	One tag at a time
Identification	Identifies the type of item (not uniquely)	Each item identified uniquely	Read/write, peer-to-peer and card emulation mode
Read/Write	Read only	Both	Both
Technology	Optical(laser)	Radio frequency	Radio frequency
Interference	Cannot be read if obstructed	Metal, liquid and other radio waves	Due to its read range, almost no interference
Automation	Mostly requires human to operate scanners	No need of human intervention	May require human intervention due to security issues
Cost	Very low	Low to high (depends on the type of tag)	Relatively low

That way, the choice of RFID technology to act as an interface to the user is an advantageous solution. It offers a clean and easy way for the user to interact with the system, in other words, the user may use the medicine cabinet as if was a normal medicine cabinet, as long as the tags (prescriptions, medicines and user cards) are within the RFID reader range.

2.2 Speech Synthesizer

A speech synthesizer is a tool that translates written text into spoken language and it is advisable to use it in four situations: when the application tasks require the user's eyes to be looking at something other than a screen (e.g. medicine package); when there are situations on which it is necessary to call the user's attention (e.g. medication reminders); when the users have certain disabilities (e.g. visual impairment due to age); and when the application attempts to have a "personality", so that the users may associate it with specific features (e.g. medication advisor). These situations, as the examples showed, may be related to the

medication process, being that the reason why a speech synthesizer was chosen for this master thesis [22].

Figure 4 below shows the steps necessary to convert text to speech. As an example, the phrase "*Leave at 5:30 on 5/15/99.*" is used as text input and the first phase is called "structure analysis", where paragraphs, sentences and other structures have their "start" and "end" determined. Punctuation and formatting elements are important on this phase. On the input example, the "." would determine the end of the paragraph, the ":" the start and end of a time, and the "/" the start/end of a date. The second phase, "text pre-processing", processes the special constructs of a language such as dates, times, numbers, email addresses, and other elements, translating them to text. By the end of this phase, the input example would be something such as "*Leave at five thirty on May fifteenth nineteen ninety nine.*"

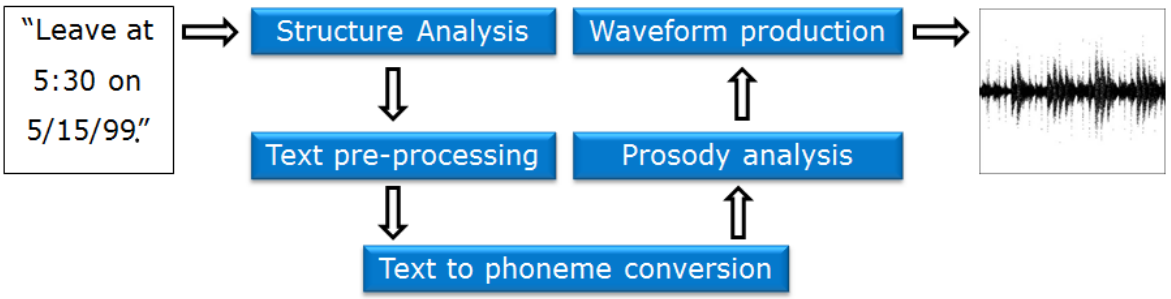

Figure 4: Speech synthesis process

The remaining phases will convert text to sound, having such process initiated with the "text to phoneme conversion" that takes every word and converts them to phonemes. The word "leave", for example, has five letters but only three phonemes. It is important to emphasize that each language has a specific number of phonemes, for instance, the word "tsunami" has a sound that does not belong to English, the "ts", but is present in Japanese. The fourth phase, "prosody analysis", combines the results of the prior phases and determines the best way that the sentence must be spoken. This includes the pitch (or melody), the timing (or rhythm), the pausing, the speaking rate, the emphasis on words and many other features. As an example, if the punctuation on the input example had a "?" instead of a ".", more emphasis would be given to the sounds on the sentence end, as it would be a question.

The final stage is the "waveform production" that uses the phonemes produced on the "text to phoneme conversion" step and combines them with the "prosody analysis" result to produce the audio waveform via two techniques: "concatenation", on which recorded pieces

of sounds from human voice are put together; or using “formant synthesis”, that takes advantage of signal processing techniques based on the knowledge of how phonemes sound and how prosody affects them. Although only five stages are necessary to convert text to speech, many tools on the market fail to accomplish two important features to which the human ears are very sensitive: understandability and naturalness.

Understandability refers to how reliably a listener will understand the words and sentences spoken by the synthesizer (e.g. if the word “cap” is clearly different from “cup”). As the naturalness feature is an indication of how the synthesizer sounds like a human voice (not all applications need that, but it is often desirable). Having the text to speech process explained, another aspect that must be taken into consideration is the variability of speech synthesizer tools that are available on the market.

If a user wants to write an application with text-to-speech capabilities, for example, the normal path would be to program specifically for the TTS (Text-To-Speech) Engine in use. To worsen things, most of TTS engines are bundled with a Recognition Engine, which enables the application with speech recognition. Thus, the same programming effort would also apply for the later engine. That way, the user has to understand the particularities of each engine and his code would probably not work with other engines.

To standardize the interface between applications and engines, APIs are provided. Even though the user now has to write code to comply with the engine API, the problem persists as each engine would have its own API. To standardize the API of most of the TTS and Recognition engines, Microsoft provided an API, which is known as SAPI. Through it, both the engines and the programmers have to comply with the same interface. Figure 5 shows the so called “Microsoft SAPI” architecture.

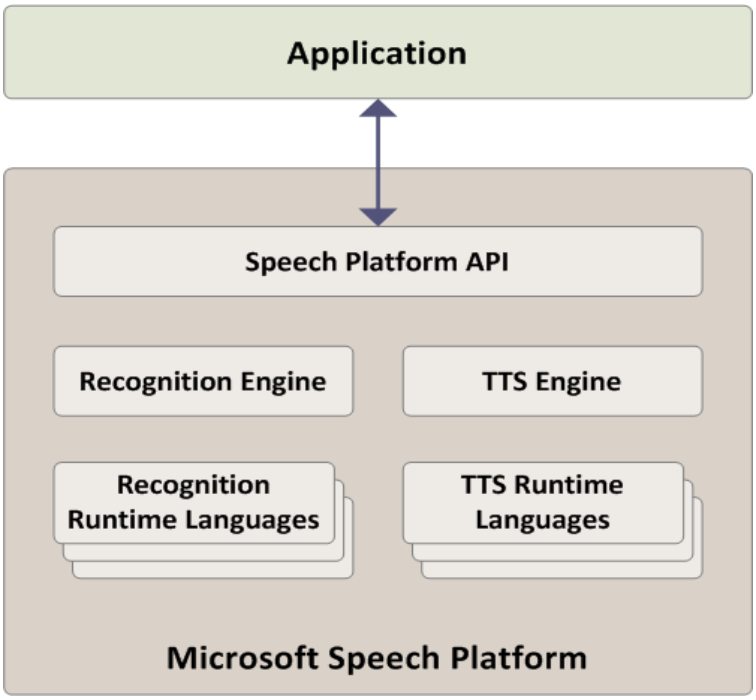

Figure 5: Microsoft Speech API architecture

The SAPI can be seen as an interface between the application, which needs text-to-speech and speech recognition capabilities, and the speech engines, which provide such functionalities. In newest versions of SAPI, applications and engines do not communicate to each other directly, but they access the same runtime component, as seen on Figure 5.

For this master thesis, a TTS Engine is necessary to instruct the user in taking medicines. The solution in the form of a prototype must only have a library which access SAPI and does not have to care about which TTS Engine that is used. As much as the Windows operating system is used, the application code of the prototype does not have to be changed if a new TTS Engine is used, let us say, if there is the need to use a synthesizer with better voice quality. The only things that have to be changed are the name of the voices used by the new TTS Engine. As it can be seen, this API decision is mostly concerned about the modularity of the system, that is, the change of one software module does not affect so much the functionalities of another.

2.3 Chapter 3 highlights

In this chapter the basic of the technologies used on this master thesis was shown so that the reader may understand how they work. First the RFID technology was described into

details on how it can be used for automatic identification of objects and why passive tags were chosen to be used rather than active ones due to their cost. A comparison was also made among RFID, barcodes and NFC technologies, which is important to clarify that there were other alternatives that could be used, but through the analysis of the impact on usability of each one, RFID was the one which best suited the purpose of this master thesis.

Finally, the process of how a text input can be converted to speech was shown. The requirement of naturalness for the voice output and the easiness on the integration of different text to speech tools via an API was determining to provide, respectively, more usability and more availability of languages to the final user. This chapter, indeed, serves for the reader to get familiar with the technologies used and as an understanding on how the choice of a technology may impact on usability and accessibility for the final user interface. The next chapter will show a list of related works that approached the problem of medication adherence; some of them used the same technologies discussed on this chapter, while others tried complete different approaches, but all of them showed a functional solution.

Chapter 3- Related Work

On this chapter, nine related works that approached the medication process are described. Although there is an extensive list of works for this area, only those that proposed solutions to increase patients' medication adherence were selected. Furthermore, the research was made focusing on the following eight factors which impact on the medication adherence, as cited in [23]: 1) low levels of health literacy; 2) poor understanding of the medication purpose; 3) lack of importance given to the impact of medication on health outcome; 4) memory or any cognitive impairment; 5) lack of a person to accompany the patient's treatment; 6) regimen complexity; 7) loose communication between patient and doctor; 8) not proper or inability to pay for medication.

Each related work will have its most relevant characteristics described and, in the end of each section, their best features are highlighted and bad ones are pointed out. These observations could be only made after all of them were described, but due to the variety of solutions proposed, it was more appropriate to list them after each section. The complete list of desirable features to increase mediation adherence, however, combining all of their features with some that are important for this master thesis, are put together onto two lists, in the end of this chapter, which will serve as a common ground of comparison.

3.1 Experiencing NFC-based Touch for Home healthcare

In [24] a health monitoring system for family members and caregivers is described. It is divided into two main parts. The first one is a health application in charge of the users' health measurements, such as weight and blood pressure. The second one is in charge of medicines taking surveillance. All the information generated by the use of such system may be accessed by a doctor or caregiver via the Internet. Figure 6 shows the overall concept.

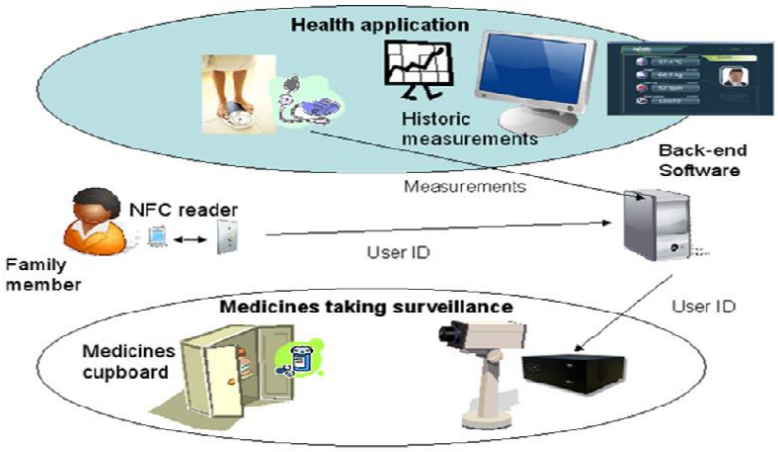

Figure 6: NFC health monitoring system [24]

The patient may use a cell phone, bracelet or card to be identified by the system, all with NFC. Once the user is identified using one of these devices, all the subsequent measurements will be assigned to the identified user. All the information generated by the use of the system is shown on a display, including the history of measurements formatted in charts for the ease of understanding.

The second part, which is important for this master thesis, is related to the medication. Once the medicine cabinet is opened, a camera starts recording the taking of medicines. The videos are stored on a backend computer and may be accessed remotely, like the measurements, by the caregiver. The caregiver is only allowed to access one patient's videos by also using NFC identification and, that way, check if the patient is taking the medicine correctly.

One problem that may arise with this application is that some users may find the video recording invasive, jeopardizing the in-taking detection if they do not allow the recording. Another point to highlight is that such detection is performed manually, increasing the caregivers' work load, which may also decrease the effectiveness if many patients have to be monitored, due the relatively low number of caregivers available on the area. Finally, a beep sound with blinking light indicate that the user should take a medicine only when they are in range and no further instructions are provided.

3.2 A Prototype on RFID and Sensor Networks for Elder Healthcare: Progress Report

A prototype which combines RFID technology with wireless sensor network to monitor elderly medication was developed in [25]. They use both HF (high frequency) and UHF (ultra high frequency) tags to identify respectivaley medicine bottles and patient. The HF tags may be read within a distance up to 30 cm and the UHF in distances from 3 to 6 meters. Besides of that, there is a weight scale to detect if a pill was taken out of the package. Figure 7 shows its architecture.

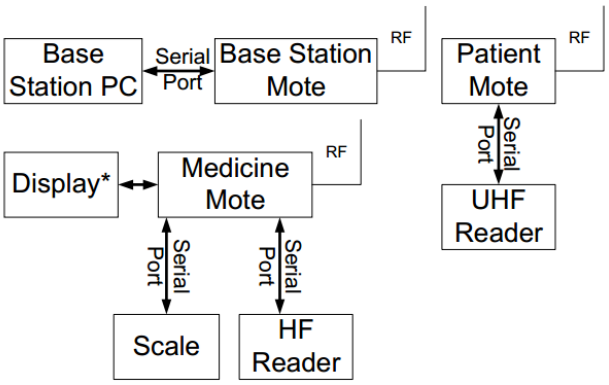

Figure 7: RFID with wireless sensor network architecture [25]

Figure 7 shows three main modules which are “connected” via RF (radio frequency). The display shows information in a GUI about which medicines the user has to take or the history of use. The scale and HF reader exchange information with a medicine mote, which represents the medicine cabinet. Their information, along with the information of the patient mote, compose a message which is transmitted to the base station mote with the following values: weight change, tag no longer detected, tag detected again, and patient detected.

In general terms, the system work as follows.When a patient gets in the vicinity of the medicine cabinet, this information is detected by the UHF reader and the patient mote sends it to the base station mote. When a medicine package is taken out of the medicine cabinet, this action is detected by the HF reader and sent to the medicine mote. When the medicine package is returned to the medicine cabinet, the data from the scale is also sent to the medicine mote. The base station mote uses the patient’s and medicine mote information, all combined in on single message, to know if the user is taking the medicines correctly. Another important characteristic happens when the user presence is detected and they have to take a

medicine (say using a prescription). In such situation, the system alerts the user to take the medicine via a beep sound or blinking light.

Again, as in [24], there is no further guidance for the user's medication besides the beep and light signals and the notifications take place only when the user is in the vicinity of the cabinet. Finally, even though the scale may be very accurate to detect when pills are taken, if more than a medicine is taken out of the cabinet and multiple pills are taken, it is not possible to be accurate about which package specifically each pill belonged to.

3.3 Interaction in Pervasive Computing Settings Using Bluetooth-Enable Active Tags and Passive RFID Technology Together with Mobile Phones

In [26] it is shown how a pervasive computing architecture may be used to interact with users using Bluetooth devices and RFID tags. They show three scenarios on which two classes of user interaction are tested, the first one on which the user initiates the interaction with a smart object; and the second one, on which the smart objects initiates the interaction with the user. Only the third scenario, which uses a hybrid interaction with the user, is considered for this master thesis, as it is a "smart medicine cabinet scenario". Figure 8 shows their prototype.

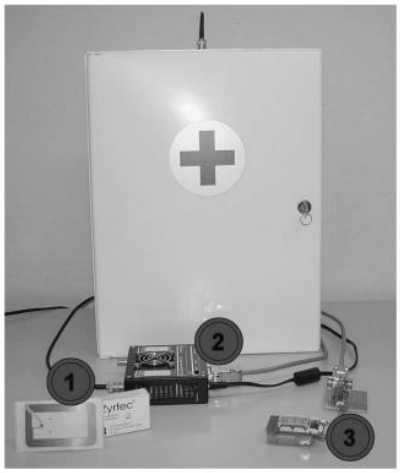

Figure 8: Smart medicine cabinet [26]

On Figure 8 there is a Bluetooth node (3), an RFID tag (1) and reader (2). RFID passive tags are placed in the medicine packages and the reader is put in the medicine cabinet. Furthermore, there is a single active tag in charge of processing the information provided by

the reader and establishing a connection with a mobile phone, via the Bluetooth node, to transmit the information collected while the cell phone was not in range and the detections while it is in range of the node.

In this scenario, the cell phone is not only used as a user interface, remote communication link and commands storage device, but also as an access point for the smart objects. That way, the medicine cabinet works in a "disconnected" way, in other words, only when there is a cell phone in its vicinity, the data is updated (history of medicine usage). Drug usage is detected by monitoring the "appearance" and "disappearance" of the packages when the patient removes from and places them in the cabinet.

The medicines tags contain medicine information, but they may also have user prescription information (shared storage). Thus, if the user has a prescription to a medicine, the package is designated only for such purpose. The prescriptions information is stored on the cell phone and is transferred to the cabinet on the next synchronization. On the other hand, the alarms are transferred to the cell phone using the Bluetooth link between the active RFID tag, which manages the cabinet, and the cell phone.

Besides of the complex pervasive environment developed, there are some possible problems of their solution. If a user takes out a medicine, but does not really take it, the way they monitor drug usage will indicate a false in-take. Another thing is that medicines may be tied to prescriptions, which may prohibit other patients from using the same medicine. Finally, the information related to the medication process are only updated when the user is in the vicinity of the cabinet, which may make the user miss a prescription schedule, for example.

3.4 Monitoring Activity of Taking Medicine by Incorporating RFID and Video Analysis

In [27] a system to detect the in-taking of medicines is developed and it is mainly intended to be used by the elderly, aiming to increase their medication adherence. They combine RFID technology and video tracking to determine whether the user has taken a medicine or not. Figure 9 shows their system overview.

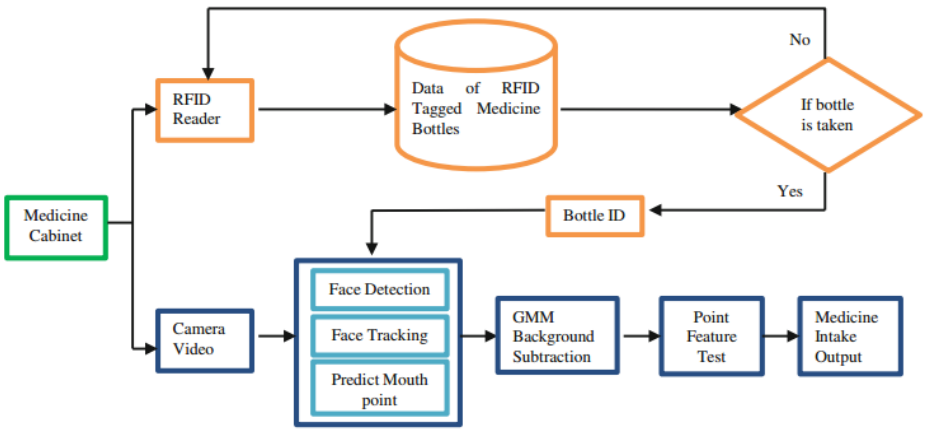

Figure 9: RFID & video used to detect medicine in-taking [27]

Although a medicine cabinet module is shown, they have not actually developed a prototype with a real cabinet, but they have used real hardware to show their idea. To evaluate the system, they have conducted experiments with 14 users handling three medicine bottles over an RFID antenna (which represents the cabinet) and having a camera positioned in front of them to detect if they have taken a pill or not.

First, an RFID antenna detects the medicine bottles which are equipped with RFID tags. As soon as they enter the read range, their IDs are used to get their information from a backend database. Whenever a bottle leaves the read range, its ID is sent to the camera module, which detects the in-taking of medicine in four phases: in the first phase, the user's face and mouth position are detected and are tracked; on the second, an image processing algorithm is used to detect movements (e.g. the arm movement towards mouth); on the third, the blobs of the image moving elements are determined (e.g. the borders of the moving hand); and the fourth calculates the distance between the blobs and the mouth position to determine if the user hand is in the vicinity of the mouth, indication that the medicine has been taken.

They say that the user is reminded to take scheduled medicines and if the user picks up the wrong medicine, they are warned about that, but they do not provide information about how these two managements are performed. Even though they say there is 100% accuracy for medicine detection by the RFID, they have used only three bottles within a very short distance from the antenna, which is not really practical for a common cabinet use. Finally, they have reported that the mouth position detection is not always accurate and that the medicine detection gets impaired if there is more than one person in the camera view.

3.5 Monitoring Medicine Intake in the Networked Home: The iCabiNet Solution

In [28] an intelligent medicine cabinet (iCabiNet) is proposed using residential networks and medicine smart packaging. The main idea of this project is to provide a way that user may be alerted about taking medicines wherever they are, be it using home appliances or remote messages on mobile devices. Figure 10 shows their idea.

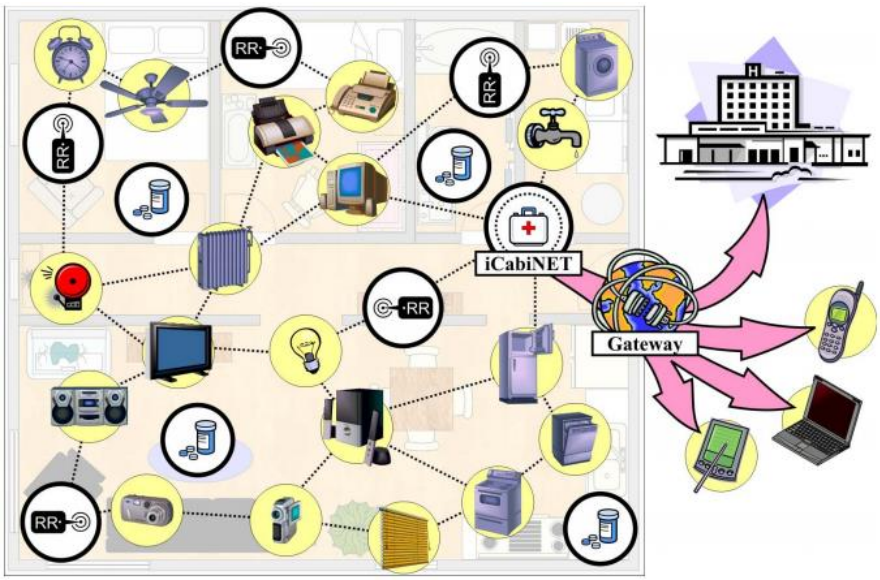

Figure 10: iCabiNet solution [28]

The iCabNet monitors the medicines usage by placing RFID tags on each pill slot on a blister pack, that way, when the tag circuit is broken (pill taken out); it is not detected anymore by the reader and considered as taken. The cabinet is then connected to a residential network; through which all the house RFID readers (RR on the figure) collected information is transferred to the cabinet. The readers are spread, so that medicine intake is detected anywhere it happens. The residential network is also used to trigger alarms to the user using any device available which is connected to it.

The other part of the system is the gateway. It enables the iCabiNet information to connect to the Internet, having its usage information shared (e.g. alarms on a cell phone) or data inputted to the system (e.g. medicine guidance). Common operations flows for each action (related to taking a medicine) are predefined on the system, while others may be downloaded by the user and a software update is necessary.

They described two complex scenarios and said their solution could realize them, but nothing was said about how each scenario is indeed solved by their purpose. Besides, few details are given about the prototype and the medicine cabinet is only represented by a Bluetooth enabled cell phone. Finally, their solution requires compatible household appliances to be connected to the residential network and many sensors (RFID readers) may be placed at home environment. Such system features undoubtedly would have a great impact on the solution cost, but the intake detection would be almost 100% precise.

3.6 .Building Common Ground for Communication Between Patients and Community Pharmacists with an Internet Medicine Cabinet

In [29] an "Internet Medicine Cabinet" is proposed. Alike most related works which use sensors and integrated hardware to monitor and support users' medicine intake, this work is mainly focused on the information involved on such process and to provide a common ground interaction among patients, pharmacists and doctors.

To achieve such common ground with success, they have focused on two classes of information to be included in the "Internet Medicine Cabinet": medications, which is the main topic of discussion and should include prescription, complementary and over the counter drugs, and their past medications; and discussion issues, such as troublesome symptoms, fears and questions, which may be strictly related to a medication or not. Through their system the user may share information related to these two classes with pharmacists and doctors.

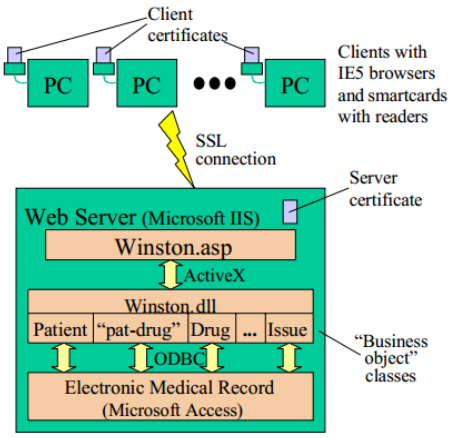

Figure 11: Architecture of a common ground for patients, doctors and pharmacists [29]

With the system shown in Figure 11 they provide a key point in the perspective of the patient, which is trust in the information-sharing relationship. Each patient, pharmacists and doctor must have a smart card with the solely purpose of access granting to the information. The patients are responsible and in total control for their own records (medication or discussion issues) and, after they use their card on a PC, such records may be sent to a database via a secure socket layer connection using a web browser. On the other side, pharmacists and doctors may access the shared information if the patients give them access (their smart card ID).

The main purposes here are to increase interaction with the three types of users; detect new drug interactions (not only prescribed ones) based on the patient's records; allow pharmacists to advise patients via the reported issues; increase the doctor knowledge about the user's medicines intakes; remembering of past actions, such as history of medications; increase patient's knowledge about medicine terms, improving their relationship with the pharmacists and doctors; and encourage them, with a reliable environment, to report more information related to self-medication and alternative treatments.

Bearing in mind the importance of a qualitative view over these pieces of information, they have also purposed an agent-based architecture to process all this information [30]. By the use of the agents intelligence, it is possible to detect patterns in the patient's behavior, providing guidelines for the pharmacist when addressing discussion issues, which improves medication guidance; the patients may be supported in understanding and management of their conditions; once a user's knowledge map is built, the system may respond to the user in a different way, according to the patient's knowledge and even detect if they suffer from a certain problem.

3.7 Multimodal and Adaptable Medication Assistant for the Elderly

In [31] a mobile application was developed to assist the elderly in taking medicines, aiming an increase in the medication adherence. The central idea is to provide an adaptive application to provide alarms and information of medications to the user. By adaptive they mean the development of a graphical interface that may resize its screen components based on the distance between user and cell phone; the flexible input as touch (using the screen) or by

speaking to the phone (using a speech recognition), that way, if the patient has rheumatism, situation on which the use of the touch is not ideal, the user is still able to use the application via voice; and noise detection, to adjust the speech output volume or gain of the speech input from the users voice. Figure 12 shows the architecture of their system.

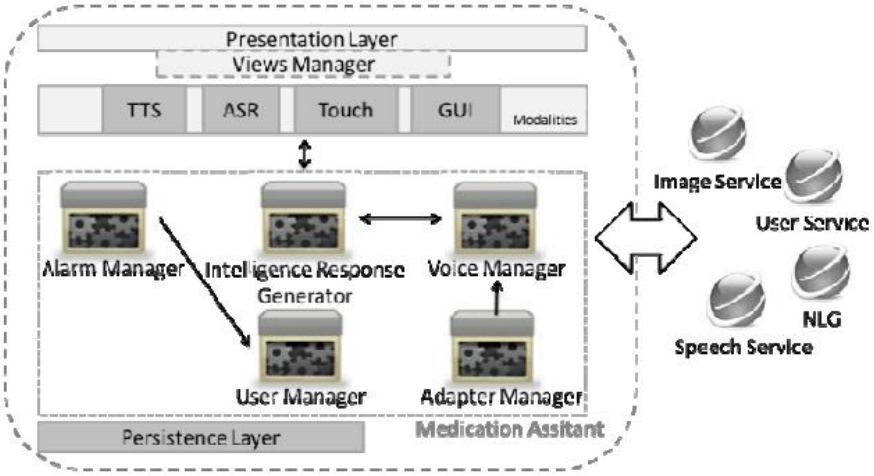

Figure 12: Adaptive mobile application to increase elderly medication [31]

On Figure 12 it is possible to see the main modules. The application itself contains a "views manager", which handles all the views; the "user manager", which handles user information and provides communication with the user service; the "alarm manager", which alerts about scheduled medicine intakes; the "voice manager", which handles the voice interactions; and the "intelligence response generator", which is in charge of interpreting the user interactions and adapting the information according to the user context.

Besides of that, there are four services to support the application. The image service generates images of packages and pills, according to each medicine characteristics. The speech service provides a speech synthesizer and recognizer. The user service, which stores and handles the user and alarms information. And the NGL (natural language generation), which plays a central role in both generation natural language sentences to the user using the database elements; or interpreting the many possible ways a user may say a sentence and translate them to an internal language, which can be only understood by the system.

As it can be seen, it is not a simple application. With it, it is possible for the user to ask the cell phone about their medication status (e.g. if they have forgotten an intake) or consult information about a specific medicine (e.g. daily dosage recommendation). Even though it is very strong in supporting the user intakes, it lacks in assuring that the user really takes the

medicines. Another weak aspect is that there is no management of medicines interactions and incompatibilities with the user.

3.8 MHS: A Multimedia System for Improving Medication Adherence in Elderly Care

In [32] there is another attempt to increase the elderly medication adherence, but the solution is based on a multimedia system combined with a context aware approach. They have taken into consideration the following requirements related to a multimedia healthcare for the elderly: “adaptive multimedia presentation”, on which a variability of ways to present information is available to the user according to which device is near to them (e.g. a cell phone next to the patient receives content designed to such device); “context-aware prompting”, that is, the system has to deliver reminders according to the patient’s current situation (e.g. intake reminder triggered only when user is likely to give attention to it); “acceptance focused on situation detection”, so that the system may determine whether the user is receptive to a certain care service (e.g. user in a situation prone to take medicines); “autonomous management of medicines”, on which the system automatically records all drug information (e.g. new medicine added); and “multiple medication taking plans”, so that patients are reminded to take a pill not only based on the right time to take it, but also based on the context detection (e.g. user must take pills after having dinner).

Taking into account these requirements, a prototype system was developed, whose architecture is illustrated on Figure 13. The “Multimedia Healthcare System” is a service-based solution and it is composed of five collaborating services: reminder, multi-medicine management, interruption detection, mediation, and presentation service.

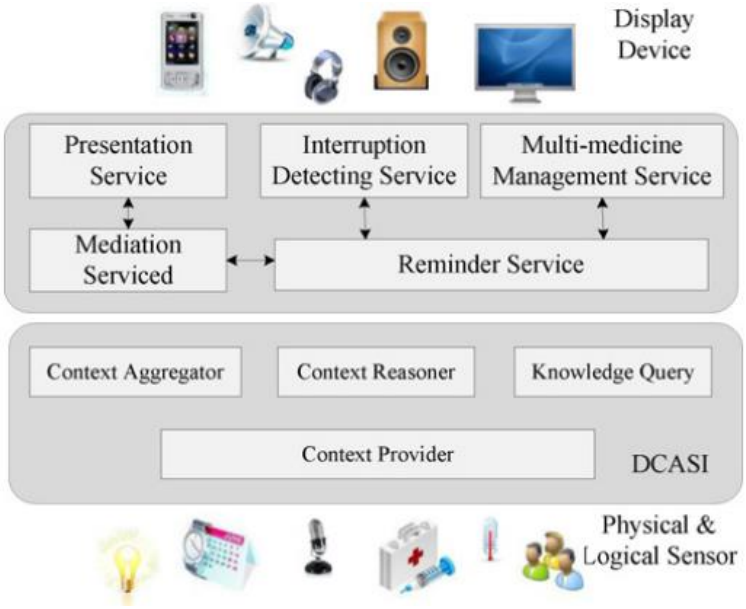

Figure 13: MHS architecture [32]

The reminder service sets a reminder whenever it is necessary, based on the medication-taking pattern. The multi-medicine management service identifies a specific pill, integrates the dosage into the medication-taking plan, and records the pill information in the database. The interruption detection service recognizes the user situation as the environment effects on how they may interact with it, measures the user acceptance using the effects, and determines whether the reminder generated is effective. The mediation service measures the display efficacy of devices, allocates the proper devices to guarantee the needs of the reminder services, and activates the right presentation services on the devices. The presentation service reports the state of devices capabilities and executes with the specific user interfaces.

They have implemented an e-cabinet, which serves as a medicine cabinet by using a RFID pad (reader) for sensing medication activity and a webcam to recognize patterns. With that, it is possible to recognize when a bottle is lifted off and put back on the pad (which means it is taken) and when a new medicine is added (user has to show the package to the webcam to store its image). To detect if a user has sat on (or stood up) a place in a room, pressure sensors are used. To detect the patient's presence, RFID readers are set on each room.

With their system the user may have an online medication schedule, recognize information related to medication, and advice on effective medication (e.g. "it is better to take a stomach pill one hour before dinner"). Beside the great complexity of their system, they still lack on the treatment of drug interactions (with others or with the user) and a "natural-like"

language to instruct the patients. Their system also relies on the sensor spreading in the home rooms, which means more costs to the end user and that if, by chance, there is a power failure on the complete system, nothing is said about how the medications states are recovered.

3.9 Feasibility Study of a Robotic Medication Assistant for the Elderly

In [33] a robot is proposed to help older users in taking medicines. Their solution philosophy, however, intends to improve the users' knowledge about medication in letting them decide which actions to take, so that they do not get dependent on their prototype. The user is identified via face recognition or ID input on a touch screen. It is mainly focused in the medicines intakes, generating reminders, logging users actions (sequence of), and generating history of use via a web service which uses the protocol SOAP to provide interoperability on data exchange. Figure 14 shows the robot used to manage user's medication.

Figure 14: Robot to manage user medication [33]

The robot is composed of a differential drive mobile platform, two single-board computers, sonar sensors, microphone, speakers, a touch-screen mounted on an actuated head, a camera, and USB ports. The mobile platform will be used as future work to allow the robot drive to the users and offer them guidance. One computer is in charge of low-level control functions (running under Linux) and the second one is designated for service applications (running under Windows XP). The microphone and camera are also meant to record the user's medicine intakes. The speakers are used by the robot for speech output (robotic voice).

And the USB ports are used to connect measuring devices used by the application services (e.g. blood pressure) and which are placed on a tray as shown on Figure 14.

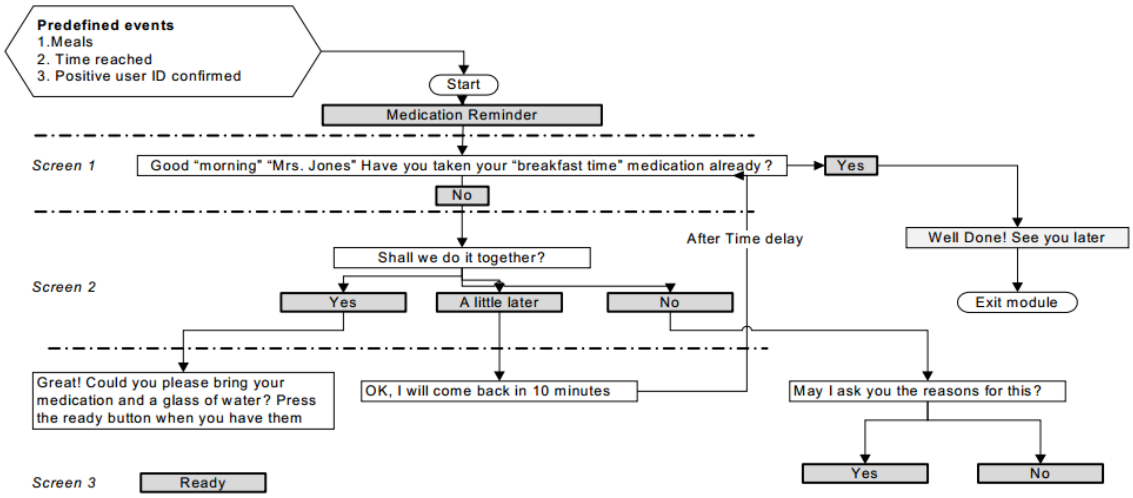

Figure 15: Robot interactive dialog system example [33]

On Figure 15 shows how the users may interact with the robot via a touch screen interface. On each screen there are at most three options available to the user. That way, the interface minimizes the complexity of actions required by the user. Another very important characteristic is that the robot gives flexibility to the user, in other words, even though the tree of actions possibilities is predefined, the sequences are chosen by the user on each interaction. Observe that it also requires the user to confirm certain pre-requisites to go from one screen to another (e.g. it asks the user to have pills and water near them to start a medication process). The screens are described using XML files and, by combining the abovementioned characteristics, they require the user to be more participative (deciding what is best for them and learning with the robot) than passive (becoming dependent on the robot management).

Their proposed system shows an intuitive user interface and has been tested with real users guided by a professional, but a further study is necessary to evaluate the impact of the system in users' daily life situations, such as: children's medication, detection of which medicine packages are handled by the patients, and normal interaction. By the later situation, it means how natural is the interaction of the system with the user in such a way that it is as close as possible to the normal life medication process without guidance and that the system is available when it is needed.

3.10 Related Work Comparison

Reading the literature it was possible to identify some related works which are relevant to this thesis. This number of works is only a sample of an extensive research that has been made to identify the factors that contribute to non-adherence, and to develop solutions to increase users' adherence to their medication. In [34] a research was made to understand what users waited from a system that makes use of in-home devices to improve medication adherence. The resulting demands, along with some relevant characteristics observed on the related works from Sections 3.1 to 3.9, are included on two tables below. These tables summarize important characteristics which may increase medication adherence. For simplicity, only "+", "0" and "-" symbols are used to represent, respectively, that a system has a positive solution for, did not approach, or poorly provided (implemented) a certain feature.

Table 2 has 15 features, which are more related to technological needs. They are described below:

- if a system can identify users, prescriptions and medicines without the patient's intervention, it is said capable of "automatic entity detection";
- furthermore, if each entity is an atomic element of the system, that is, one element may not be used to represent another one, then there is a "unique entity detection";
- the feature "many items at a time" refers for the system capability in managing the multiple addition and removal of entities from the system domain, in other words, its capability in managing multiple entities activities at the same time;
- if there is a method to detect, without prompting the user, that a medicine has been taken, the system is capable of "automatic intake detection";
- to avoid that a patient takes a wrong medicine, for example, it is important that the system computes entities relationships in a correct way and provide a "fast response";
- if the system provides interoperability, so that at least some of its functions are presented in other devices, it is said to be a "multi-device" solution;
- if notifications to the user are clearly classified into groups (e.g. dangerous situation and confirmation of positive user's action) to facilitate patients' comprehension about the system, it is said to have "message categorization";

- if not only visual signals are used for these messages, but a clear voice output with speeches addressing directly the patients is given, the system is said to have “speech capability”.
- “recall management”, which refers to the capability of the system in providing a way to warn the user about new recalled medicines;
- “medication status recovery” is the ability of the solution in providing a continuous medication process, that is, if there is a power shortage, which is very common in Brazil, the medication schedules should not be lost;
- if various users may use the same system, it is capable of holding up “multiple users”;
- for the medication, if there is almost no difference between using a system process or the normal life activity of taking medicines, the solution is said to provide a “natural interaction”, that is, the user does not have to change much on its daily life to use the system;
- the later feature is closely related to “understanding of technology purpose”, as the simpler the technology interaction is, the easier it is for the user to understand what is the technological goal;
- if, for example, new medicine interactions are found, the system should incorporate such “updated data” somehow;
- finally, it may be important for the solution to make the system usage (e.g. medicines taken, delays in medication) availble through the internet.

Table 2: Technological related features

	3.1	3.2	3.3	3.4	3.5	3.6	3.7	3.8	3.9
Automatic entity detection	-	+	+	-	+	0	0	-	-
Automatic intake detection	-	+	-	+	+	0	0	-	-
Many items at a time	-	-	-	-	+	0	0	-	-
Unique entity detection	-	-	-	-	+	0	0	-	-
Fast response	-	0	-	-	+	-	+	+	+
Multi-device	0	0	-	0	+	0	0	+	+
Speech capability	0	0	0	0	0	+	+	0	+
Message categorization	0	0	+	0	+	+	+	0	+
Recall management	0	0	0	0	0	+	0	0	-
Medication status recovery	0	0	0	0	0	0	0	0	-
Multiple users	+	+	0	-	0	+	0	-	-
Natural interaction	+	+	0	+	+	-	+	+	+
Understanding of technology purpose	+	+	-	+	+	+	+	+	+
Updated data	0	0	0	0	+	+	+	+	-
Information sharing	0	-	-	-	-	0	0	-	-

Table 3, however, is more related to the user and is also composed of 15 features, which are described now:

- on Table 2 it was shown that “information sharing” may be available for some systems, that way “privacy” must be a concern (e.g. users may not like their medication on the screen of a TV when they have visitors);
- if a system provides a way that children may be accompanied by an adult to take medicines, than it is concerned with “usage by children”;
- furthermore, if “only allowed users” are permitted to take medicines using the system, misuse of it is avoided;

- if a system is concerned with “user incompatibility”, it takes care of allergies and medicine interactions with the user;
- on the other hand, “medicine incompatibility” refers to the medicine interactions (e.g. two medicines may not be taken together);
- when a system manages expiration date and remaining quantity of pills of a medicine, it is concerned with “package status”;
- if there is a scheduled intake, the patient is reminded once, they take the medicine, but the reminders persist, the system does not have an “efficient reminder”.
- if a solution is concerned about “prescription complexity”, it will abstract the difficulties of the prescription regimens (e.g. prescriptions that are not taken every day) and the user has only to pay attention to the reminders;
- sometimes when the patient forgets to take a medicine on the right time and, by the time they remember, many hours have been elapsed, doubts may arise, that way it would be important for a system to manage “schedule tardiness”;
- a system should also provide “medication flexibility”, which is the feature that allows a patient to start a medication when they wish and, if in case of forgetfulness, there is a window time on which the medicine may be taken without problems;
- in order to avoid overdose, a system must take care of “medication dosage” in such a way that the daily dosage and the allowed dosages for each user age group is maintained;
- “users engagement” is a desired feature because it means how much a user may enjoy using a system as a means of taking their medication;
- the latter feature may be increased if it is provided a way to switch the speech language easily, taking into consideration the “user’s language”;
- for a person who lives alone or for the elderly, the capability of a system in having “interaction with caregiver” (message exchange) may have an important role on the patient’s treatment;

- finally, if a system manages “usage history incompatibilities”, than it manages if a user tries to take a medicine, but it is incompatible with a previously taken one.

Table 3: User related features

	3.1	3.2	3.3	3.4	3.5	3.6	3.7	3.8	3.9
Privacy	+	+	+	+	0	+	+	+	+
Usage by children	0	0	0	0	0	0	0	0	0
Only allowed users	+	0	0	0	0	+	0	0	0
User incompatibility	-	0	0	-	0	+	0	0	0
Medicine incompatibility	-	0	0	-	0	+	0	0	+
Package status	0	0	+	0	+	-	0	0	0
Effective reminder	-	-	-	-	+	-	+	+	+
Prescription complexity	0	0	0	0	0	0	+	+	+
Schedule tardiness	0	0	0	0	0	0	0	+	+
Medication flexibility	0	0	0	0	+	+	0	+	+
Medication dosage	-	-	-	0	+	+	+	+	+
User engagement	+	+	-	+	+	+	+	+	+
User’s language	0	0	-	0	0	0	0	0	0
Interaction with caregiver	-	0	0	0	-	+	0	0	+
Usage history incompatibilities	-	0	0	0	-	+	0	0	0

3.11 Chapter 3 highlights

The reader may be thinking about why so many features were listed, but one must not forget that the medication is a process strictly tied to the user and, that way; it involves many subjective aspects that, most of the time, cannot be properly measured and must be analyzed in a more qualitative form. While the 30 features presented on this chapter are important to see the challenges that have been faced by many solutions which attempted to increase medication adherence, none of the related works handled, at least in a basic form, all of them.

Bearing in mind all the features discussed on this chapter, the reader is now ready to go to the next chapter, where the medication process will be analyzed into more details and a solution will be presented in an attempt to involve all of the abovementioned 30 features. Furthermore, in the conclusion of this document, the two tables listed on this chapter will be very important to verify how this master thesis approached each one of items and how better the solutions were when compared to the related works.

Chapter 4- Conception of an intelligent medicine cabinet

This chapter is divided into two parts, which are the description of the problem that this master thesis intends to solve, and the solution thought for it. The first part will condense the variables involved in the medication process into three main entities (user, medicine and prescription) and the relationships between them are shown. Those relationships are the key factors to understand the problems involved with medication adherence.

On the second part the proposed solution of thesis is presented: the intelligent medicine cabinet. On such part, it is important to recover those thirty items (Table 2 and Table 3) discussed on the last chapter to realize the importance given by this master thesis to the state of art during the conception of its solution. Finally, each one of the relationships between the entities is explored to help in understanding how the intelligent medicine cabinet is supposed to increase medication adherence.

4.1 Baseline Situation

To better visualize the concept of the intelligent medicine cabinet, it is first necessary to understand the baseline situation of the medication process. On the related work it was possible verify the existence of some important elements involved on such process: the young and older groups of patients, the use of medicine packages or bottles, and medication schedules in the form of reminders. All these elements, however, need only to be represented by three entities that are the user, medicine and prescription illustrated on Figure 16.

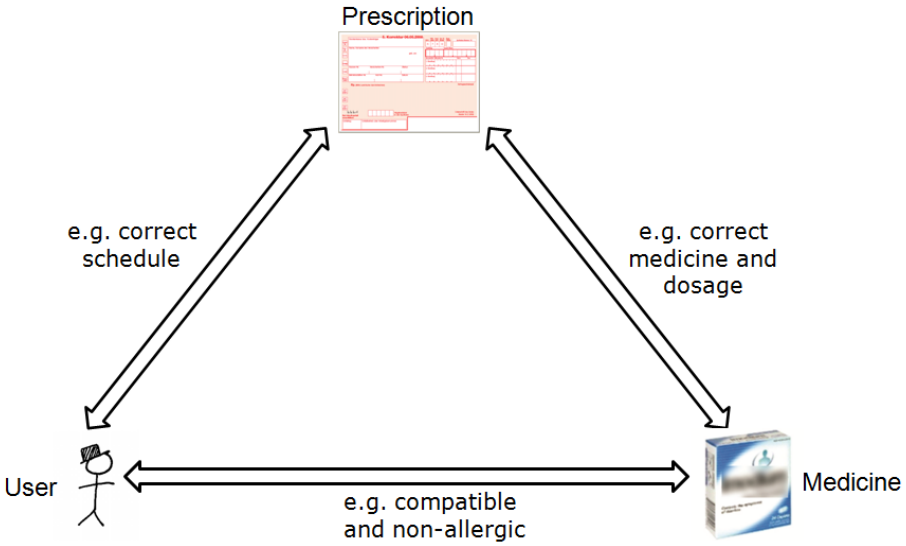

Figure 16: Normal medication baseline

Figure 16 shows that these entities may have specific relationships such as “correct schedule” between the user and prescription; the “non-allergic” property between the user and a medicine; and what is the “correct dosage” of a medicine for a prescription. These relationships, however, are only a sample of many others that were identified by reading information about many medicines in a “therapeutical records” source, such as U.S. Food and Drug Administration [35] or ANVISA [36], reading a book [37] and consulting medical staff. The most relevant ones, which are most common, were chosen and are described below.

Between the user and medicine entities, it is important to consider allergies, incompatibilities with an active ingredient or active ingredient group of the medicine, and if the package is not expired, empty or has been recalled. In addition to that, the recently taken medicines data are important to avoid mixing incompatible ones in case their effect is not over on the user’s organism and to assure that the daily dosage rate is obeyed, avoiding overdoses.

As for the relationships between user and prescription (which also involves the later with medicine) the frequency on which the drug is taken (e.g. every eight hours), the right dosage for each schedule (e.g. two pills per intake), and the total duration of the medication (e.g. take a medicine for one week) are important aspects. Finally, for the user, the right time to take the medicine is essential.

One of the typical issues that might come up with the medication schedules is that the user might miss the right time to take a medicine. What should be done upon this schedule

tardiness is not a trivial solution. With the sources mentioned above, it was possible to identify the following general scenarios for a delayed medication:

1) The missed schedule should be skipped and the medicine should be taken on the next schedule, as previously set. The missed schedule should be added to the end of the total medication period.
2) The user should take the missed schedule as soon as possible and continue the medication as regular.
3) The missed schedule should be taken immediately and the remaining schedules should be rescheduled based on the newly taken time.
4) The user should ignore the missed schedule and take a double dosage on the next schedule.

Which action should be performed and what is the allowed delay for the schedules, so that there is no damage for the user, depend on each medicine characteristic. These pieces of information may be found on their leaflet or, in the last case, the physician (doctor) should provide them on the user's prescription.

As can be seen, there are many variables to be considered when taking medication. As cited in [23], most of the time, the user does not know all of this technical information (low level of health literacy), neglects them (lack of importance given to the impact of medication on their health), or finds the medication process tedious (due to regimen complexity). On the other hand, with the system proposed on this master thesis, the person does not have to handle all these parameters alone, as there is a unit to manage them. Figure 17 shows the simplified concept.

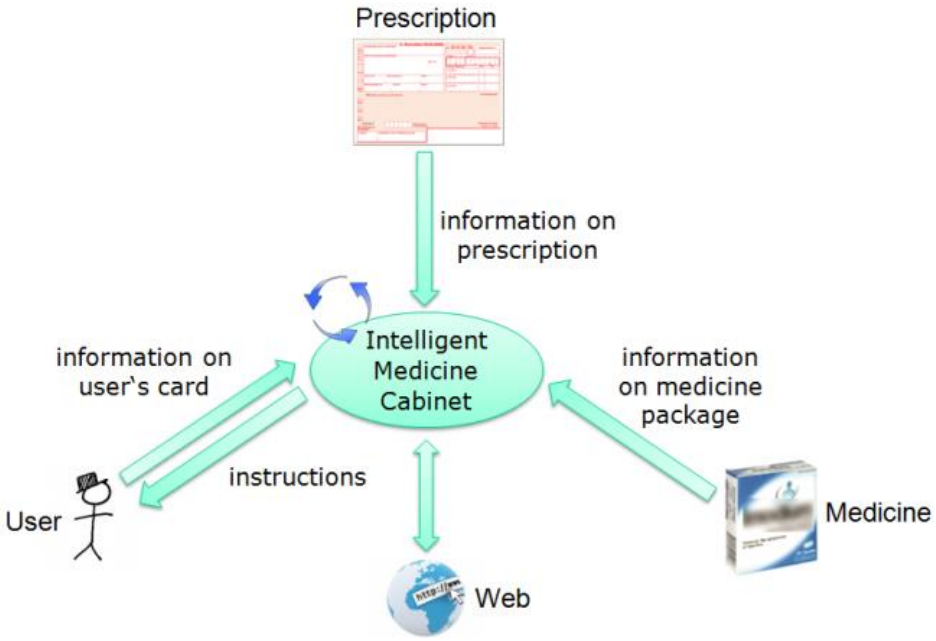

Figure 17: Intelligent medicine cabinet concept

Figure 17 shows that the user does not have to interact with the other two entities directly. That way, even though the user might have poor understanding of the medication background or suffers from a kind of memory or cognitive impairment [23], the handling of all the relationships between the three entities are left to the intelligent medicine cabinet in an automatic fashion. Besides of handling their relationships, the solution may also store the user's history of usage and make this information available to the web. Next section shows the concept in an extended way.

4.2 The Proposed Intelligent Medicine Cabinet

Figure 18 shows the extended version of the intelligent medicine cabinet concept of Figure 17. On this version the figure, the doctor is present to show that the web interface is not a mere channel to make the cabinet usage history available to the web, but also an interface through which the doctor may add new information (e.g. new medicine interactions) to the solution and interact with the user (e.g. send them a message). The information shared on the web, however, must have the user's agreement for privacy reasons.

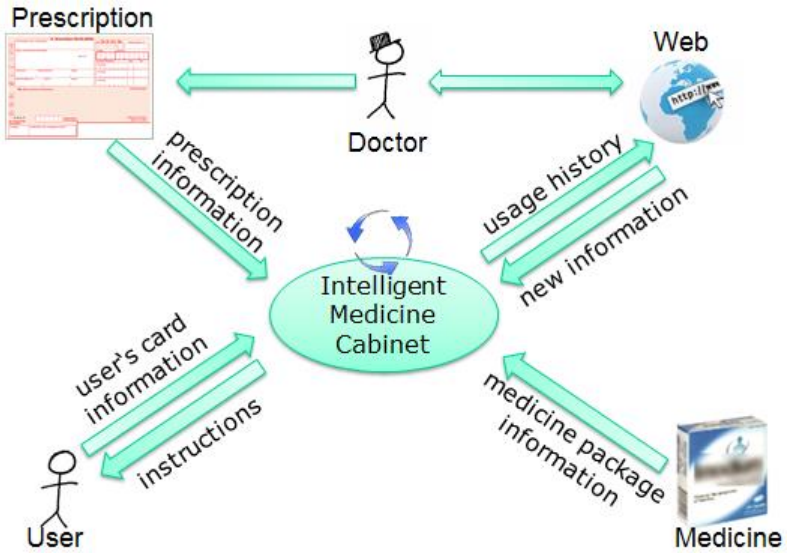

Figure 18: Intelligent medicine cabinet extended concept

The intelligence of the system begins with the automatic detection of users, medicines, and prescriptions information as soon as they are within range. Based upon these entities relationships, the proposed system may help patients in taking the right medicine and on the right time, instructing them (if possible, speaking their language) and generating reminders. Furthermore, it registers which medicines are taken by whom, at what frequency and time; making this information available via a web service and allowing the doctor to monitor the patient's progress.

By the use of a technology that can get information of the entities wirelessly, tags must be attached to them, providing a way to store their data. Given that, the user must have cards with their personal information. On the back of the prescriptions, tags with details about the medications are attached; and inside the medicine boxes, tags are placed with specific information of each package. To properly interface with the medicine cabinet, the information of these tags must be previously saved to them (prescriptions should be saved by a doctor).

The solution may manage two types of medicines: over the counter and prescription ones. As the complementary medicines are not officially advisable by doctors, they are excluded from this master thesis. In addition, the concept of the intelligent medicine cabinet classifies prescriptions on whether they allow repetition or not. That is necessary as some prescriptions need a pause between medication cycles, but with the intelligent medicine cabinet, the user does not have to handle all the burden of such complex regimens.

Once the entities information is available, the user may use their card to log in the system. When a user logs in, they become the active user, in other words, all the actions

performed after such login are considered as made by the active user, as much as other users do not log in afterwards. Being an active user is a pre-requirement to use the cabinet, which allows the user to open the cabinet and, consequently, to place medicines and prescriptions into it, as well as to remove and take them out.

The user's identification is necessary so that only allowed people may use the cabinet. If a person tries to use it without a valid ID, the concept does not allow further interactions and warns the user about it. This feature is also relevant for children, as they may only use the medicine cabinet if an adult accompanies them.

If a new prescription is added to the cabinet, its information is automatically read and the patient may start taking the medications on its list. As for the medicine, when it is added, if it is expired or has been announced as dangerous or been recalled, the user is immediately informed of that. As can be seen, the data interactions among the entities must be computed efficiently and provide a fast response to the user, so that the intake of an expired pill, for example, does not happen.

To take a medicine, the person needs only to take it out of the cabinet, the intelligent medicine cabinet associates the active user's ID with it, and if there is any reason for which it should not be taken (e.g. ingredient incompatibility), the cabinet informs of that. Otherwise, after some seconds, the drug is automatically considered as taken, may be put back into the cabinet and the user is informed of the remaining quantity of pills if only few are left. Being informed of the remaining quantity of pills (and expiry date of a package) is important to avoid surprises when the user needs to take a medicine.

After considered as taken, if the medicine is under prescription, its medication information is updated on the cabinet (time taken, if there was any delay, and next schedule). In case a medicine should not be taken, when it is returned to the cabinet it is not considered as taken as the patient was warned about the danger. It is important to emphasize that the system assigns priority for prescription drugs. Therefore, if the user tries to take a medicine (not a prescription one) that is incompatible with a prescription in use, such action is denied and a warning is generated.

With the concept of an active user, many users may be logged in the cabinet, but only one may use it at a time. That is necessary so that the medicine cabinet detects who is currently using it. With that in mind, when someone logs out, their session is closed and the previous logged in user becomes active (if any). But, even though a user may have their session closed, medicine schedules are not lost. In other words, the user does not have to be

logged in to be informed about medicines under prescription, but they will be announced on the right time. These aspects show that the cabinet is able to manage many items at a time (prescriptions, users' cards, and medicine packages). This is possible due to the nature of the solution, which treats each action separately, verifying the interactions of the involved variables at the moment of the action.

Furthermore, a web service fulfills the abovementioned callback function, in case a drug is newly announced as dangerous or its batch number has been listed with a manufacturing fault. It also provides statistics of usage history of the medicine cabinet, allowing the users' treatment to be monitored by a doctor or a caregiver chosen by the patient. It is also possible to insert new information into the medicine cabinet via this web service (e.g. new medicines interactions and messages from the caregiver) or use it to provide some of the medicine cabinet functionalities to other devices (e.g. a mobile application that uses the cabinet information to generate reminders when the user is not in the vicinity of the cabinet).

The cabinet also provides flexibility by allowing the user to choose when to start their medication or when to stop it, in case the medication causes any inconvenience (e.g. strong side effects). For each schedule, periodic reminders are triggered for a pre-defined period of time. If the user, however, takes the medicine before such period is over, the reminders for that schedule are removed. Furthermore, if the medication is stopped (i.e. prescription removed), no schedules are calculated for it anymore unless the prescription allows repetition (i.e. medication restart). The user is also instructed about what should be done in case the right time to take a medicine is missed. The medicine cabinet allows one prescription to have two medicines on its list and may manage more than one prescription at the same time.

The proposed concept also includes the management of permitted dosage rate, which is the allowed quantity for each age group or, for a period of 24 hours, it is not possible to take more pills than the recommended daily dosage for a specific medicine. Furthermore, there is the management of user's allergies, medicine incompatibilities (with user or other medicines), and of medications under power constraints. The later refers to the medicine cabinet capability in recovering the status of a medication even after a power shortage takes place.

4.3 Chapter 4 highlights

Even though medication may be thought as a simple activity, as it is something completely common in people's daily lives, the first part of this chapter showed that, if no proper care is given, it may become dangerous to those people. It was not only important to show how the baseline of the medication process is, on which the user has to deal alone with many parameters involved on the medication, but also the characteristics that may turn it into something complex. With a clearer view of the problem, it is now possible to understand why medication adherence is a great challenge in developing solutions for it.

On the second part of this chapter, the proposed concept by this master thesis was presented. Along the text, twenty seven aspects of the previous chapter were discussed along. The remaining three are left to be discussed here to show the importance of this chapter for the rest of the document. By analyzing the stated concept, it is possible to see that little effort from the user's side will be necessary to use the proposed medicine cabinet (i.e. natural interaction) and, given its simplicity, it will also be easier for patients to understand the technology purpose involved and be more engaged on what really matters: their medication. Finally, with the understanding of the problem and introduction of the proposed concept, the reader is now prepared to go to the next chapter, where it is shown how the problem was modeled.

Chapter 5- System Development

This chapter shows how the intelligent medicine cabinet was in fact developed. The system use case and architecture are shown. The use cases describe the possible ways through which a user may interact with the intelligent medicine cabinet and each one of them will be explored into details. After that, a discussion about which type of design, functional or object, best suited the development of the proposed system.

The architecture explanation is divided into groups to describe into more details the following subjects: the “entities data abstraction” is made taking into account the attributes of the entities that are most relevant to the medication process; the “RFID reading performance” topic points solution to issues related to such technology; the “cabinet notifications” section shows the system IO messages; the “program logic” explains how the system works; and the “web service” shows how it is possible to share (and get) medication information through the web.

5.1 System Use Case

As stated before in the conception (Chapter 4-), the intelligent medicine cabinet may be used by the patient and a doctor. For simplicity (and flexibility) its use case diagram is shown here with a general user. Most of the use cases are intended for the patient, but a few of them are mostly intended for a doctor. Bringing together the possible ways this actor may interact with the system, some uses cases were created and are describe below.

As part of the *“place prescription”* use case, we have that as soon as the prescription is added to the reading range by the user, it is checked whether it is new or was previously present on the system. If it is new, its information is saved and the user is informed that they may take a medicine from its list, otherwise, there are two possibilities according to the time elapsed since the last time it was taken out of the cabinet: in case of a short duration of time out of the cabinet, nothing happens when it is placed back; if it was long enough to be considered as removed, it is checked if the prescription allows or not medication repetition. Does it allow repetition, the medication may be restarted and the user is informed that the

prescription is valid again, does it not, the user is informed that the prescription is not valid again.

The *"remove prescription"* use case refers to the taking out of a prescription. When a user takes out a prescription of the cabinet for a short period of time, it is only considered as taken out and may be placed back into the cabinet without any change on the medication, otherwise, if it is away for a longer period of time, it is considered as removed, in other words, the prescription is flagged as inactive. If it allows medication repletion, the next time it is placed back into the cabinet it will have its medication restarted, otherwise, it will not be valid anymore and another prescription is necessary for such medication.

For the *"place medicine"* use case, when a medicine enters the reading range, it is considered as placed into the cabinet and the user may be informed of three situations: that the medicine has been taken, a new medicine has been added or the medicine was not taken. The first case happens when the package had been taken for a time enough to take a medicine and is extended by the use case *"take medicine"*. The second case happens when it is checked that the package has not been previously added and is extended by the use case *"add new medicine"*. And the third case happens when the time was enough for the pill to be taken, but it was checked that there was any incompatibility for which it would be harmful to the user to take it.

The *"add new medicine"* use case allows that after the medicine has been placed into the cabinet and it was checked that it was not present before, the system searches for information about such medicine on its database, so that when the user takes it out, it is possible to check all the incompatibilities.

For the *"remove medicine"*, after a medicine leaves the reading range of the cabinet, three cases may happen based on the time it takes for the medicine to be placed back again: if it remains removed for a short period of time (not enough to take any pill) nothing happens; if it remains out of the cabinet for a period of time enough to take the medicine, the use case *"take medicine"* is used; and if it stays away of the cabinet for a longer period of time, the use case "*remove medicine from system*" is used.

The use case "*take medicine*" happens after the medicine has been taken out for a period of time enough to consider it has been taken, so the system checks if the user may take it and, in positive case, it is registered on the system that it was taken, regardless of being a medicine with or without prescription. If it is with prescription, of course, the related information, such as next schedule, is also updated. On the other hand, if it is verified any incompatibility of the

medicine with the user, they are informed of that along with the reason. This use case extends both the use cases *"place medicine"* and *"remove medicine"*, as it depends on both to happen.

The *"remove medicine from system"* happens if a medicine leaves the reading range for a long period of time (more than it is necessary to take it), so it is considered as removed from the cabinet and the user is informed of that, having the medicine information deleted from the system memory.

The use case *"log in"* is a pre-requirement to use the cabinet. It happens when the user places their card within the reader range. After a login, the user actions may be extended by the following use cases: place medicine, remove medicine, open cabinet door, become active user and log out.

For the *"open cabinet door"* use case, when the cabinet door is open it is checked if there is someone logged in the system, if not, the user is warned to not use the cabinet. If there is someone logged in, verification is necessary: if the logged in user is a child and there is not any adult logged in along with. If there is no adult logged in, the child is warned not to use the cabinet without proper supervision. In case the logged in user is an adult or a child accompanied with one, after the door is open a welcoming message is spoken to the user along with positive visual signals.

The use case *"become active user"*, as the name suggests, makes the user become active after two situations: he was the last one to log in the system or he had logged in just before the current user who has just logged out. It is possible to see that the system queues the logins in a LIFO (last-in-first-out) style. The active user is important for the system to detect who is currently using it, otherwise, if there was not such delimitation, it would be impossible to know who is currently interacting with the system and ambiguities could be generated.

The use case *"log out"* happens when the user cards leaves the read range, having the user logged out. This use case is extended with *"update active user"*. As for the *"update active user"*, when a user logs out, it is checked who was the previous other user to log in, if any, and that one becomes active.

The use case *"request information"* is made via the web service and may be used by the doctor to monitor the patient's progress. It is extended by *"request statistics"* and *"request operating status"*. By a set of services of the system web service, it is possible to check the history of usage of the cabinet, in other words, it is possible to monitor the patients' medications progress (e.g. if the medicines are taken on the right time), covering the *"request*

statistics" use case. As for the *"request operating status"*, via the web service it is also possible to know if the intelligent medicine cabinet is operating properly.

Finally, with the use case *"add new information"*, it is possible to add information necessary for the proper operation of the system. It is extended by *"add tag information"* and *"add database information"*. For the first, by using a GUI interface it is possible to store data to the entities' tags. This interface may be used, for example, by the doctor to store the necessary information to the patient's prescription. As for the second one, a set of operations of the web service allows the addition of new medicine information to the system. It is possible to add new drug interactions, general medicine data or recalled package information.

The use case diagram of the system is shown in Figure 19. The arrows with *"uses"* represent the *"include"* relation. Except from the *"add new information"* and *"request information"*, in all the other use cases the user is informed via visual signals and voice output of what the system is doing. That is important so that the user gets a feedback of the system that each one of their actions is being detected and processed by the intelligent medicine cabinet. Otherwise, as an example, the user would place a medicine into the cabinet, it could not be detected and, as there is no confirmation of the action, the user would not know the package was not added to the system.

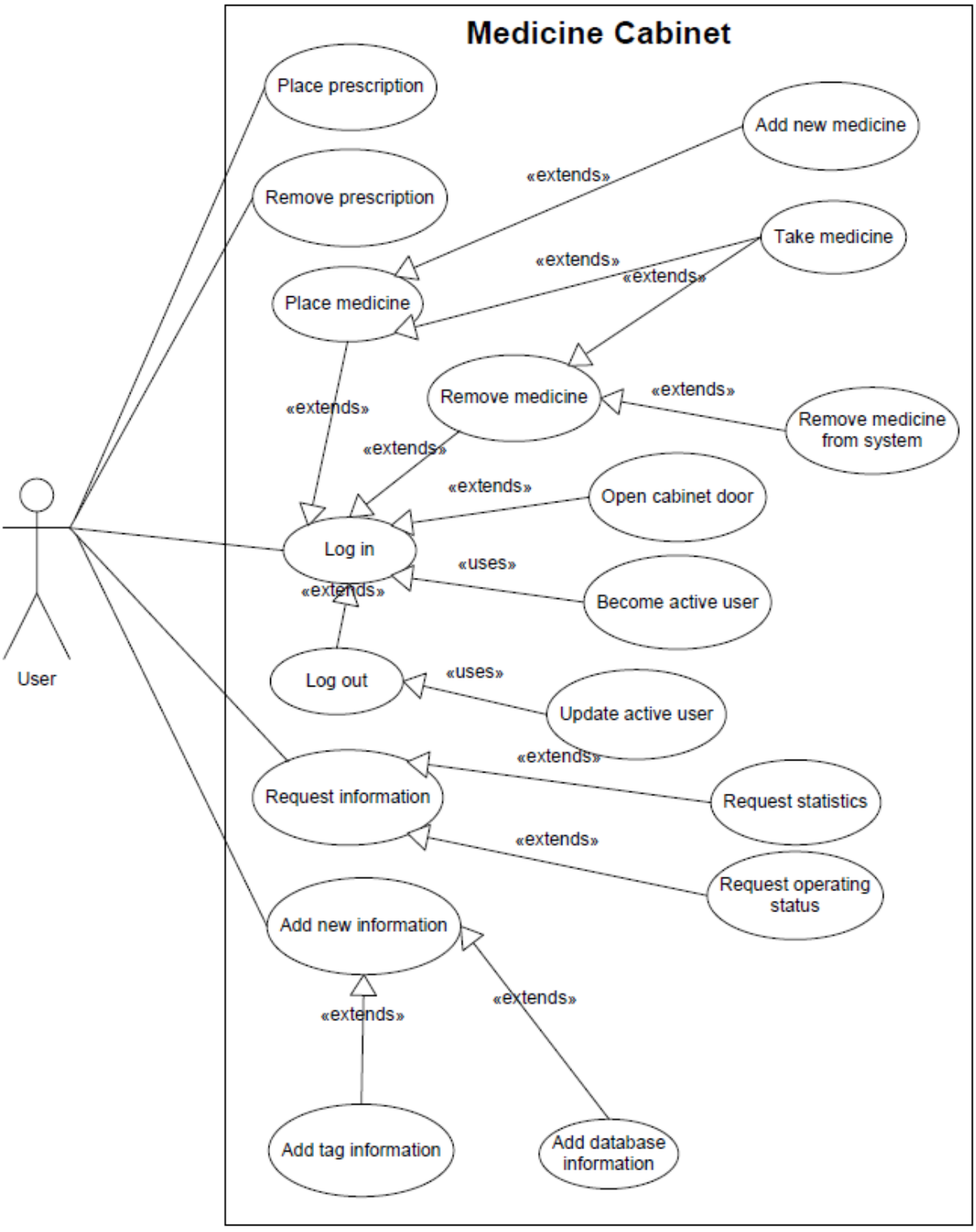

Figure 19: Intelligent medicine cabinet use case diagram

5.2 Functional Abstract Design

When a system is on its definition phase, it is defined what it should look like. That is, what its requirements and model should be. But nothing is said about how the system should be implemented. That is exactly the main purpose of the design phase, in other words, of how the functionality of the system is going to be realized. With a consistent architecture, it is possible to show the system structure, which components compose it and the interfaces among them [38].

While the definition phase is task or problem oriented, the design phase is focused on its realization and implementation (this chapter). As a medicine cabinet has the entities user, medicine and prescription, one should expect that an object abstraction analysis be made. But, as the system should provide much functionality, a functional abstraction analysis could also be used.

In an object oriented design the modules of the system represent data abstraction, while in a function oriented design, each module represent a functional abstraction. Independent of the method used, it should be possible to decompose the system into smaller pieces, each piece having its own functionalities separate from each other. But that does not happen in reality. Due to system complexity, sometimes, each piece should at least communicate to each other, sharing some features. That is where abstraction should be used, as it allows that the components be analyzed in terms of their services (or behaviors) provided to the user, with no need of considering too many implementation details [39].

Bearing that in mind, it is possible to use an object oriented design, but, instead of using data abstraction, the services provided by each module should be more relevant. That way, each object would represent not only an entity in a bundled data concept, but also its services (or behaviors). That makes possible a functional abstraction for an object oriented implementation and it makes easier the modularization of the system, which implies in easier maintenance and understanding of it.

This method to model a system was applied to the Intelligent Medicine Cabinet. Although it seems to be a very basic design, it was of fundamental importance to model a complex system such as the one proposed on this master's thesis and guarantee flexibility and modularization of its entities and components.

5.3 Prototype requirements

Since the first prototype of the intelligent medicine cabinet was developed in Germany and they have a more practical approach about development, the requirements for a prototype are already defined in the beginning of the master thesis and can be used later to verify if the research accomplished what it proposed in the form of a prototype. Therefore, taking into account the concept proposed on this chapter, a list of requirements is already presented below.

General functional requirements:

- The system should have a professional speech synthesizer integrated into the system to generate speech instructions for the user;
- The medicine cabinet should be able to connect to local network and provide information over the internet;
- A light system should be designed to provide useful visual signals for each system announcement. The concept should categorize all announcements and provide a light signal to each category;
- The system should be designed to avoid hazardous usage by children and only allowed user IDs should be able to use it normally.

Web based functional requirements:

- The system should have call-back function implemented. That is, it should manage information about newly dangerous announced medicines;
- It should be possible to query the statistics (history of usage) about the medicine cabinet. That way the doctor could monitor the patient's progress;
- The medicine cabinet should provide an update function, in which information about the medicine could be changed or added;
- The system should provide an interface, through which other devices could collect information like the statistics or receive alarms about the time the user should take medicine.

Logic functional requirements:

- The time on which the medication starts should be flexible. The user should have the possibility to choose when to start it;
- The medicine cabinet should consider the user's language when interacting with the patient;
- The medicine cabinet should remind the users about the time a medicine should be taken even if the user is not logged in;
- If the user does not take medicine on the right time, the system should send alarms for a predefined period of time;
- The medicine cabinet should manage if the person is allowed to permitted dose rate for adult or child;
- The system should manage user's allergies and medicine incompatibilities;
- The system should manage information related to expiry date of medicine packages;
- The system should manage the remaining quantity of medicament of the medicine;
- The medicine cabinet should be able to manage two or more prescriptions at the same time;
- The prescriptions could have two medicines to be taken on their list;
- The medicine cabinet should tell the user which user is currently active. That is necessary after a user is logged in or out;
- The medicine cabinet should provide the possibility that a user had also two or more prescriptions;
- The users' treatments should not be stopped unless it is indicated by the user. In other words, if there is a power shortage, the system should manage to continue its users' medications when its operation is back to normal.

5.4 System Architecture

Figure 20 shows the proposed architecture of the system.

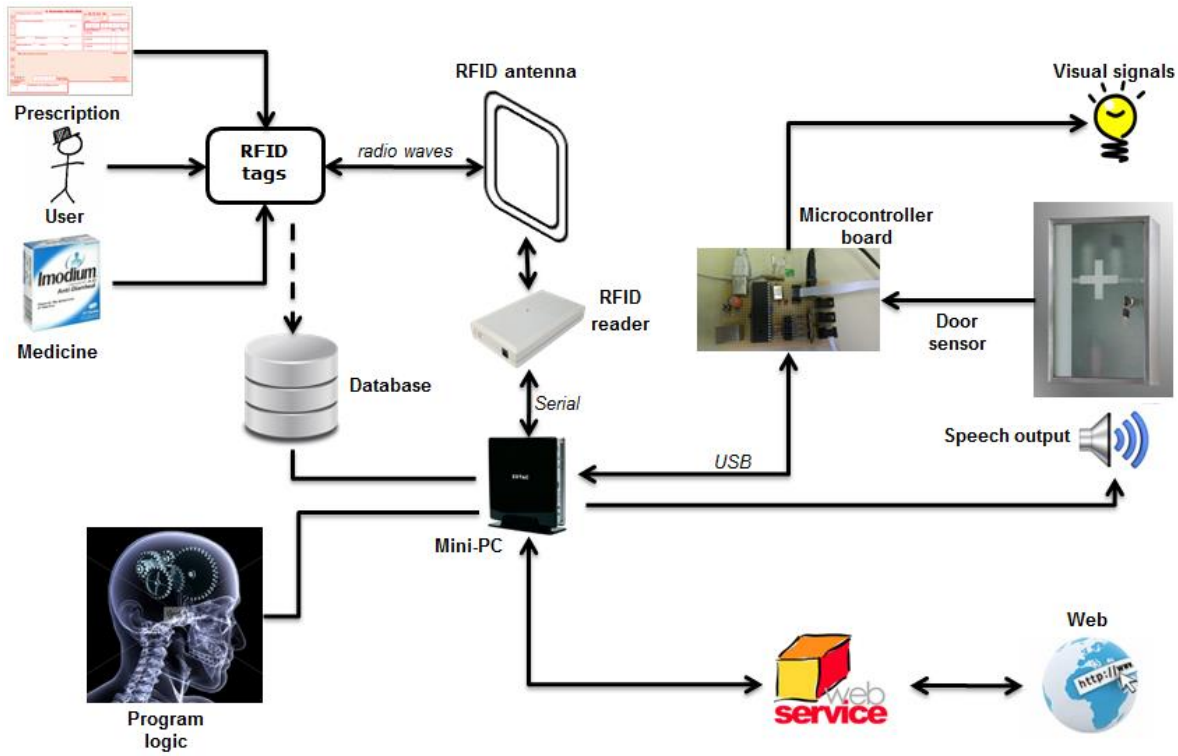

Figure 20: Intelligent medicine cabinet architecture

On the left top of Figure 20 there are the three entities of the system: user, prescription and medicine. Their information is stored on the RFID tags and, by radio waves, they are sent to the RFID antenna connected to the RFID reader. After this information is extracted from the tags, it is also stored in the system database.

There is a PC which plays a central role on the system. It is where the main program logic runs and to where all the components of the system are connected. The PC is connected to the RFID reader via serial interface and to the microcontroller board (LEDs & contact sensor input) via USB. Furthermore, there is also a web service which runs on the PC and provides information to the web.

The architecture may be divided into four major parts:

- Entity data abstraction – composed of the entities user, prescription, medicine and of the database of the system.
- RFID reading – composed by the RFID reader and its antenna.

- Notifications – composed by the microcontroller board, door sensor, speech output and visual signals.
- Web service – composed by the system database, PHP script and JSON; it provides an interface to access information of the medicine cabinet via the web.
- and Program Logic – written in Java programming language, it provides a smooth and guided medication to the user.

Each group will be described on the following sections.

5.4.1 Entities data abstraction

For the proper identification of the three entities mentioned above (i.e. user, medicine and prescription) it is necessary that they have their information stored somewhere, either attached to them for wireless detection, or stored on a database for a more detailed information. Both cases are proposed to be used for storing the result of the entities data abstraction. Generally speaking, for the wireless detection, RFID tags will be used, as they may be incorporated into the entities easily (see Figure 21) and simple information may be saved to them. As for the database, it will contain more complex information (e.g. prescription status).

Figure 21: Adhesive RFID tag

Table 4 shows that the entity “medicine” has its information grouped into generic and specific information. The generic information contains the attributes which are common to all the medicines and may have some of its attributes changed according to researches made in the medical field. The specific information contains attributes which are specific of each medicine package and may have its attributes changed according to the patient’s usage (e.g. quantity of pills).

Table 4: Medicine data abstraction

Medicine information	
Generic	**Specific**
Name	Name
Active ingredients	Batch number
Active ingredient group	Expiry date
Daily dosage	Quantity of pills

As discussed above, there are two storage units: the general information is saved in the database of the system and its content is mainly taken from the medicine leaflet, but it can also be taken from the internet, by reading "therapeutical records" [35] or updated when new medicine interactions are found or a medicine has been recalled; the specific part is stored in the RFID tag that is placed into the medicine package.

On Table 5 the user's information was also separated into two groups: personal and history information. The personal information contains the attributes which define each user individually. The history information is related to the actions done by the user to take medicines, be it using prescription or not (e.g. list of medicines taken and their respective times).

Table 5: User data abstraction

User information	
Personal	**History**
Name	Medicines taken
Age	Time take
Language	Dose taken
Allergy	
Incompatibility	

The user's history information is saved in the database as soon as they use the cabinet, but their personal information is stored on a plastic card that has an RFID tag inside it. The card may go into the user's pocket, as it carries personal information that is interesting only for the owner. There are two important data fields of the user information that are important

and need to be clarified: allergy and incompatibility. If a user says they have allergy against a medicine, than the name of the medicine is used to fill such field. In addition, if the user cannot take the medicine because of an active ingredient or active ingredient group of it, than the user is considered as having an incompatibility with the medicine.

Table 6 shows the prescription data representation. The specific information is related to the attributes of each medication, while "history" is the way to monitor the progress of the medication. The prescription medication history information is saved in the database, but its specific data is stored on the RFID tag that goes attached to the doctor's prescription.

Table 6: Prescription data abstraction

Prescription information	
Specific	**History**
User's name	Start of medication
Prescription reuse	Schedules
Medicines to be taken	End of medication
Duration of treatment	
Frequency of intakes	
Quantity to take	
Dose to take	
Delay action	
Acceptable delay	

As the prescriptions have more complex information than the other entities, due to the relationships with the user and medicine at the same time, it is important to describe its data abstraction:

- User's name – it is self-explanatory, but a user may have only one prescription at a time.
- Prescription reuse – this field tells if the prescription has repetition enabled or not, in other words, the prescription could have its medication paused for a while and, after some time, could be restarted. If a prescription is to be repeated, all the medicines on its list will have their medication restarted.

- Duration of treatment – that is the quantity of days during which the medication should be taken.
- Quantity to take – this field tells how many pills should be taken on each dose.
- Dose to take – this field does not mean the quantity of pills to be taken. As an example, if the user has the value "2" for quantity to take and he has to take double dose, that means, he should take 4 pills.
- Delay action – that means what the user should do if they forget to take one schedule on the right time. This field was already discussed on section 4.1.
- Acceptable delay – this field tells how long an intake is acceptable to be delayed, before having to do one action specified in "delay action".
- Start/end of medication – as the name says when a medicine with prescription was first taken and when the medication finished, be it because the user took the medicine properly or not (e.g. patient may decide to stop a medication because of side effects).
- Schedules – the schedules contain information about when and how the medicine should be taken. In other words, the right time and how many pills to take.

In Chapter 4-, it was shown some basic scenarios about the entities interactions. Now they had their description expanded and it is possible to see that a simple medication activity may become complicated if most of its parameters are put into issue. That way, when an entity tag is taken out or placed into the medicine cabinet, the system should react to this event very fast to check all the possible interactions of the data abstractions showed in this section. Putting everything together is real challenging and it is necessary a reliable and fast solution to detect these events. In the upcoming section the RFID solution for this challenge is described.

5.4.2 RFID reading performance

As the RFID technology plays a central role on the automatic detection of the three entities involved on the medication process, a detailed analysis of its usage boundaries is necessary. The next subsections discuss the problems that may arise during the RFID reading

process and, after that; two scenarios are shown as probable solutions, but only one is chosen due to its better performance.

5.4.2.1 Reading problems

Every technology, even though very robust, is prone to errors. With the RFID technology it is not different and the possibility of getting an error increases as a user interacts with the system. In [40] it is said that "the users work task oriented and not function oriented". Thus, they do not need to know how the RFID reader detects and retrieves information from the tags, not even what the limitations of the technology are. That way, as the user interacts with the medicine cabinet, the chance that they may set the system in prone-to-error situation increases.

During development it was verified another factor which increases the chance of the system in getting into an erroneous state: even though the RFID reader may be fast enough, it gets limited by how the operating system handles a serial communication. That way, it is not possible to make requests to the reader faster than how the operating system sends and receives serial data.

Before talking about the challenges faced, it is necessary to give a short introduction of how the reader works (based on Section 2.1). The process of getting information from the tags happens in two sequential phases: the reader must first send radio waves to detect which IDs are within its read range and then, in a second phase, request data from each detected ID. It is seen that there is a time slice between detecting IDs and actually reading their data. That is exactly where the user interaction may insert errors into the system:

- timing – if the user places a medicine, card or prescription in the reader range and its ID is detected, but when the data reading process is started, the user removes the tag from its range, there is a data inconsistency. In other words, the ID was added but there was no information about it.
- positioning – when a user places a medicine into the cabinet, for example, during the positioning movement if the medicine tag antenna is for a period of time at ninety degrees with the reader antenna, the tag might not be energized by the radio waves and will not return its data, even if its ID was detected. When that happens, the tag would be considered as removed.

- interference – if the user utilizes any device next to the cabinet which produces radio waves with frequencies equal to or close to the frequency on which the RFID reader works, there might be interference. It can also be caused by a tag not positioned properly (noise signal). This interference might, for example, cause the reader to get corrupted data and even not recognize any tag in its range for a period of time.

As it is seen, the interaction with the user may lead the system to an error state and the prototype must provide a huge range of functions, which are usually cumbersome to people when done alone [41]. By providing a solution, in which the users naturally interact with the system, as if they were using a conventional medicine cabinet, the usability of the system is largely increased. Any user can intuitively use the medicine cabinet without any technical backgrounds. On the upcoming section it is shown how these three problems were solved, because a solution was necessary for a bigger problem: no acknowledgment of tags removal or placement at all.

5.4.2.2 Dedicated reading process

An important aspect on a system is its performance in responding to the user interactions and serving them. If the user interacts with the system and requests something from it, the system should respond in time or at least respond the request, not ignoring them. The main way the user may interact with the medicine cabinet is using the tags attached to the medicines, prescriptions or user cards. This interaction is acknowledged by the removal or placing of these elements on the cabinet, as the RFID reader detects their presence or absence when reading their IDs.

If a medicine is removed from the cabinet, for example, the user should get information about it, such as: if the medicine is not expired or empty or if the medicine may or not be taken by this user. If another medicine is taken while all this information is being processed and requested from the system, the system should acknowledge this new removal and, as soon as that the previous processing is finished, search information for the new action made by the user (the second removal).

As can be seen, it is possible that actions performed by the user be not acknowledged by the system in case the system is busy with previous actions, which may be harmless to them if

a wrong medicine is taken without any warning being generated. That happens because of two operations that take time to be performed:

- The system gets busy requesting information about the detected tag and crossing its information with other entities on the system or;
- The system gets busy waiting for voice outputs to be processed and spoken to the user as messages.

Regarding user interaction, the system may be simplified into four big pieces of software:

- RFID reader module – this module is in charge of reading the tags, that is, detecting their presence or absence within the reader range, as well as saving these changes to posterior processing.
- Message output module – it is in charge of processing output messages to the user. These outputs are in the form of LED signals or of voice, which had text as input.
- Information module – it is in charge of making database queries, in other words, getting information about the medicines, prescriptions and user cards.
- Main Module – as the name says, it is the main part and it is in charge of making the medicine cabinet tasks to be performed in the right order, scheduling when each module should be called or accessed.

With that information, the understanding of Figure 22 in UML 2.0 notations is straight-forward, which shows an example of user's actions not being acknowledged. On the figure it can be seen that the user removes a medicine tag and later reinserts it into the system or even adds and removes a new prescription without the system detecting them, which is caused by the use of only a thread ("process") that has to process all information. Following the sequence on Figure 22 of the system actions: it requests the RFID reader module to retrieve tag information; checks for inventory modification (tags added or removed); requests available data from the information module; if the tag has invalid data, the main module is informed and has little to process, informing the user within a short time duration (message to the user); otherwise, if it is valid data, the message output module has more to process. This "if case" is seen on the "alt" frame of the sequence diagram.

The problem with this approach is that the system may get busy for a long time, neglecting the reading process of the tags. On Figure 22 the user removes a medicine and later

adds it again, without the system detecting this action. In that time the user may take a wrong medicine without being warned by the system.

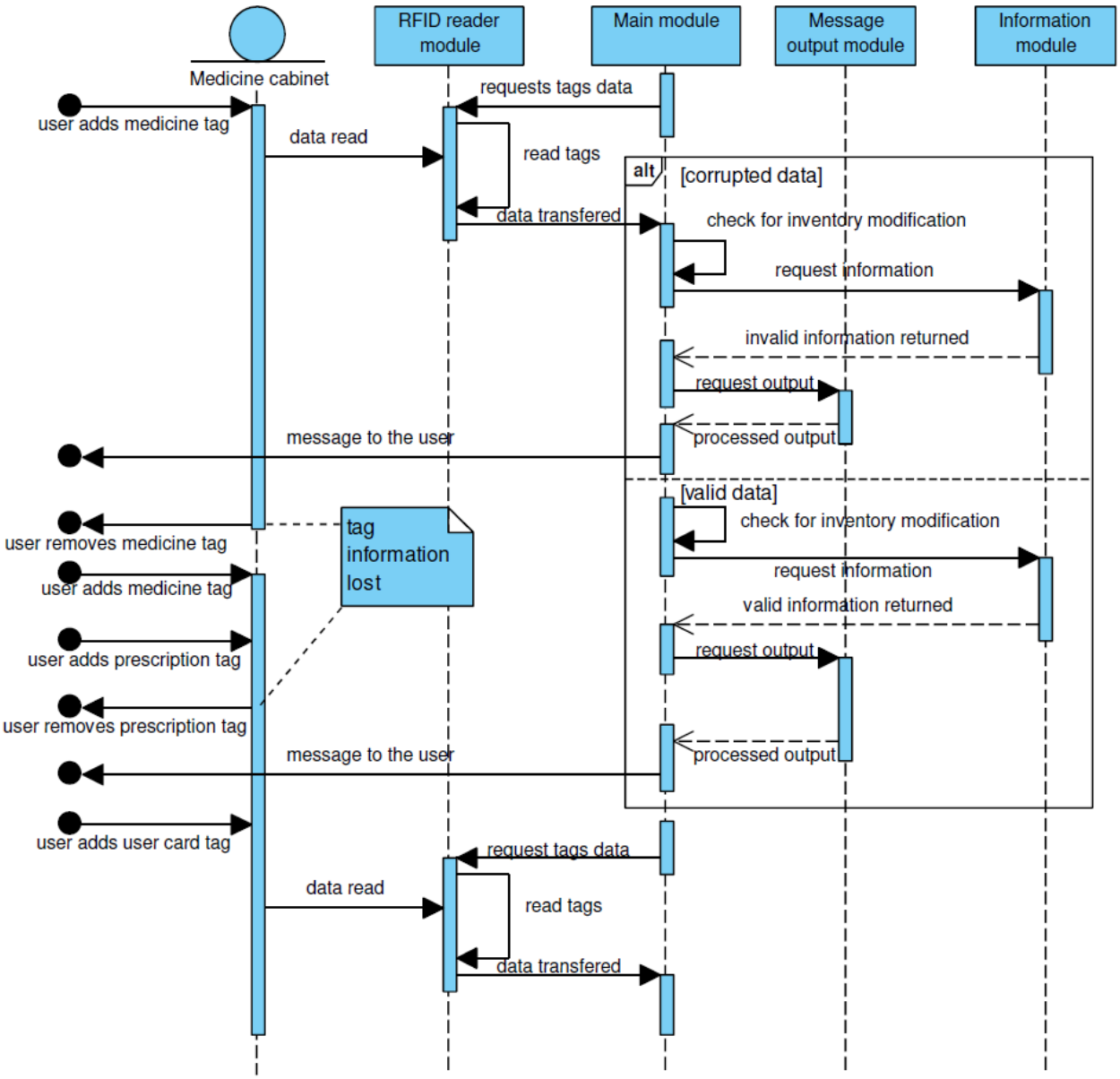

Figure 22: System sequence diagram with single thread

To improve performance, another strategy is suggested, which can be seen on Figure 23. The keyword for this solution: a multi-thread environment. A thread is the smallest and single unit of processing which can be schedule by an operating system. On the figure it is possible to see three of them: one for the tags reading process (RFID reader module), another for the main module and a third for the voice outputs (message output module). Using this approach, the changes made by the user are not lost because of the following aspects:

- Dedicated processes in parallel – even though there is a main process to manage the others, each one is in charge of specific tasks, reducing the load on one single process. That way the main module thread is only in charge of managing the processing and requesting of data from the information module, being not busy for a long time with voice output anymore, for example.
- Reading buffer – a dedicated reading thread allows the use of a buffer to store the changes made on the system until the main thread is ready to process them ("gets reader data" on Figure 23).
- Message output queue – the main module can add messages ("request output" on Figure 23) on a FIFO (first in first out) queue to be processed by the message output module. These messages are processed one after the other by this thread, eliminating the need of the main module to wait for them to be outputted.

That way, all the actions performed by the user are acknowledged by the system, as it is possible to see on the "modification detected" arrows pointing to the loop frame of the RFID reader module on Figure 23.

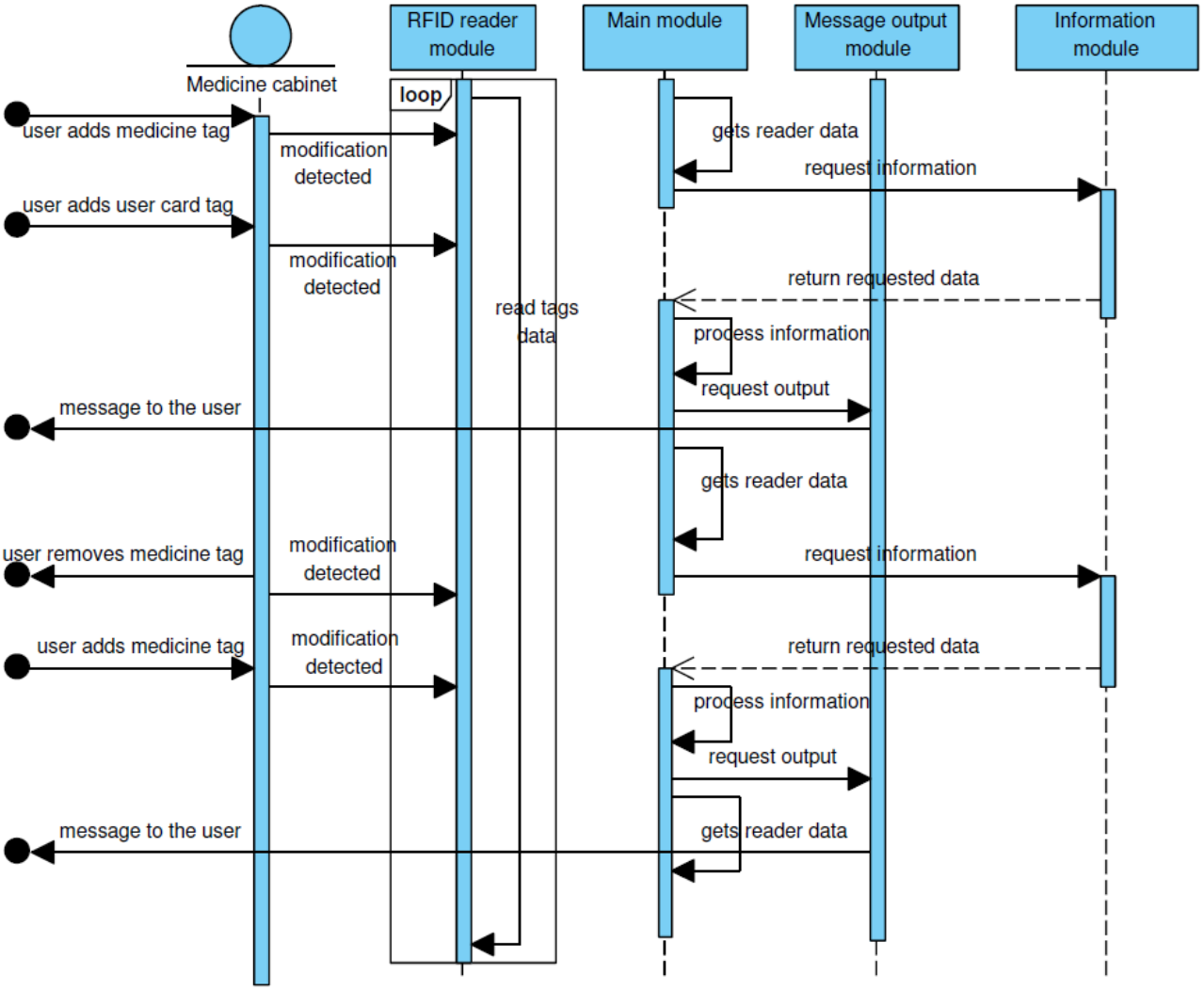

Figure 23: System sequence diagram with multi-threads

So now, with a dedicated thread to observe the tags, it is possible to get persistent reading upon erroneous responses and the three challenges described before have a possible solution:

- Timing – to solve this problem, the intelligent medicine cabinet consider the tag as removed from its range after it tries to get the tag data and nothing is returned for a certain quantity of trials in a sequence.
- Positioning - when that happens, the tag would be considered as removed. But the medicine cabinet waits a short period of time before actually considering it as removed. That period of time is enough for the medicine to be properly placed into the cabinet

- Interference: To overcome this problem, the medicine cabinet only considers all the tags removed after they are not recognized after a certain quantity of reading trials.

One may think that the second approach is the most logical, but it was not used on the previous version of the Intelligent Medicine Cabinet. Instead, the first version was used and led to the problems described above. As the medicine cabinet needs to have a high performance for its quality requirements, the second approach (multi-thread) was designed to be used for this master thesis.

5.4.3 Notifications

For the user be instructed in taking medicines, the medicine cabinet has to provide a way to give instructions and be notified about the user's activities. The main interface to detect user's activity is by the use of RFID, but there are another three that are important and are described on the next sections.

5.4.3.1 Visual signals

If the user takes a medicine and it should not be taken, the medicine cabinet should generate a visual signal corresponding to this event. The same should happen if the user does a correct action; a visual signal should confirm that. Furthermore, if the user should pay attention to something being processed by the cabinet, another specific visual signal should be provided.

Bearing that in mind, a simple way to categorize these visual signals would be using LEDs and assigning a scale of "meaning" for them. A green light categorizes "allowed actions" (e.g. patient may use the cabinet after a login), "confirmation" of actions (e.g. a medicine should not be taken and the user returned it to the cabinet), and "useful messages" (e.g. the time to take a medicine has been reached). A white light indicates that the user may pay more "attention" to a message or that "information is being processed", which is important to let the user know that the system recognized their action.

The third category, using a red light, represents the bad messages. On this class, it is possible that the patient have a "denied action" (e.g. a medicine should not be taken because of an incompatibility) or there was an "error" on the system (e.g. it was not possible to

retrieve data from the database). As it can be seen, there is a scale from “green” to “red” messages, in other words, from signals that indicate good to bad actions.

5.4.3.2 Voice instructions

Once the user is logged in in the intelligent medicine cabinet, their language is detected using the information read from their RFID card. It is checked if their language is supported and if it is, the instructions are translated to the corresponding language and the synthesizer for that language is initiated. If the language is not supported, the English language, as default option, should be initiated. Figure 24 shows how the medicine cabinet instructs the patient using their language when available.

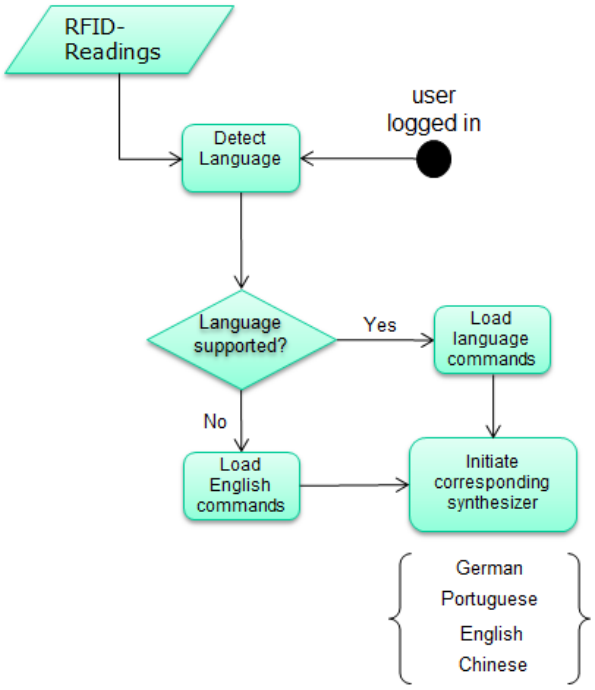

Figure 24: Voice Output

The “application” from

Figure 5 is represented here by the green flow chart. As it was discussed on that section, the other part where the languages are depicted (Figure 24) belongs to the SAPI and the synthesizer engine. There is no need of changing the green blocks (medicine cabinet application program) if a new voice is added, for example. With that, modularity is gained.

5.4.3.3 Door sensor

Figure 25 below shows the importance of having a door sensor for the medicine cabinet, which is the detection of the user's intention in using it. If the medicine cabinet did not have this contact sensor, the only way to detect if someone was using the cabinet would be via the user's card, which is not practical, as it is possible that someone without a card tries to use the medicine cabinet. That way, when the door is open, if the person does not have a valid card, the person is warned that they are not authorized.

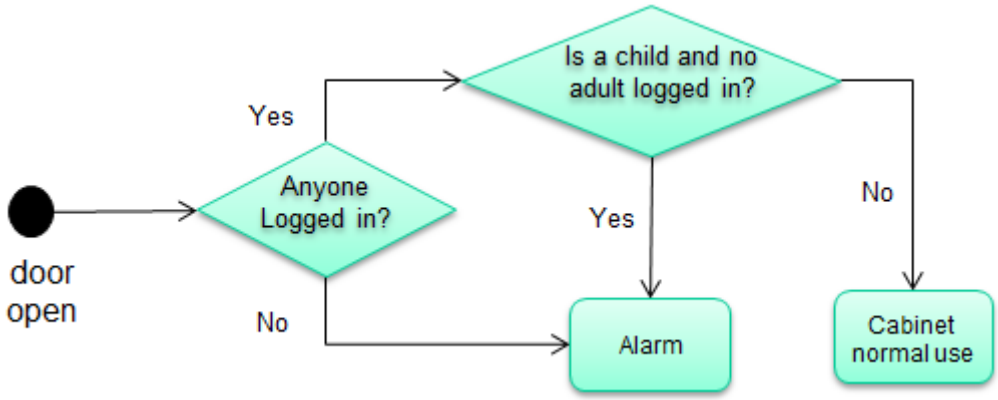

Figure 25: Door sensor functionality

The same logic applies when a child logs in, but there is no adult already logged in. Actually, a child is already warned not to use the cabinet if they are the only one logged in the system. In case the door is open, a second warning is generated, but this time along with an alarm to call attention of any adult at home.

With that, only authorized people may use the intelligent medicine cabinet normally and unauthorized ones are warned about the risks of taking medication without a valid card. It is important for the intelligent medicine cabinet to allow only authorized people because their identification is the key factor to support them in taking the right medicines and on the right time.

5.4.4 Intelligent medicine cabinet logic

Figure 26 shows the flowchart of the main source code of the intelligent medicine cabinet. It is only shown a simplified version of what happens when a medicine is taken out, so that it is possible to understand its main flow. Otherwise, if every element of this diagram

was expanded, the purpose of understanding what the system does could be compromised. A very detailed explanation of each part of the main source code can be found in the prototype chapter.

Figure 26 shows that the system performs a double checking to determine if the medicine needs a prescription to be taken, and, if not, if there is a prescription available for it. Such checking is essential to know if it is necessary to search or not for specific hazards: an over the counter medicine must have much more incompatibilities checked (e.g. allergies) than a prescription one, as the latter is already the result of a doctor's guidance (e.g. only package attributes must be checked, such as its expiry date). For both types of medicine, the user's history is an important aspect to be taken into consideration.

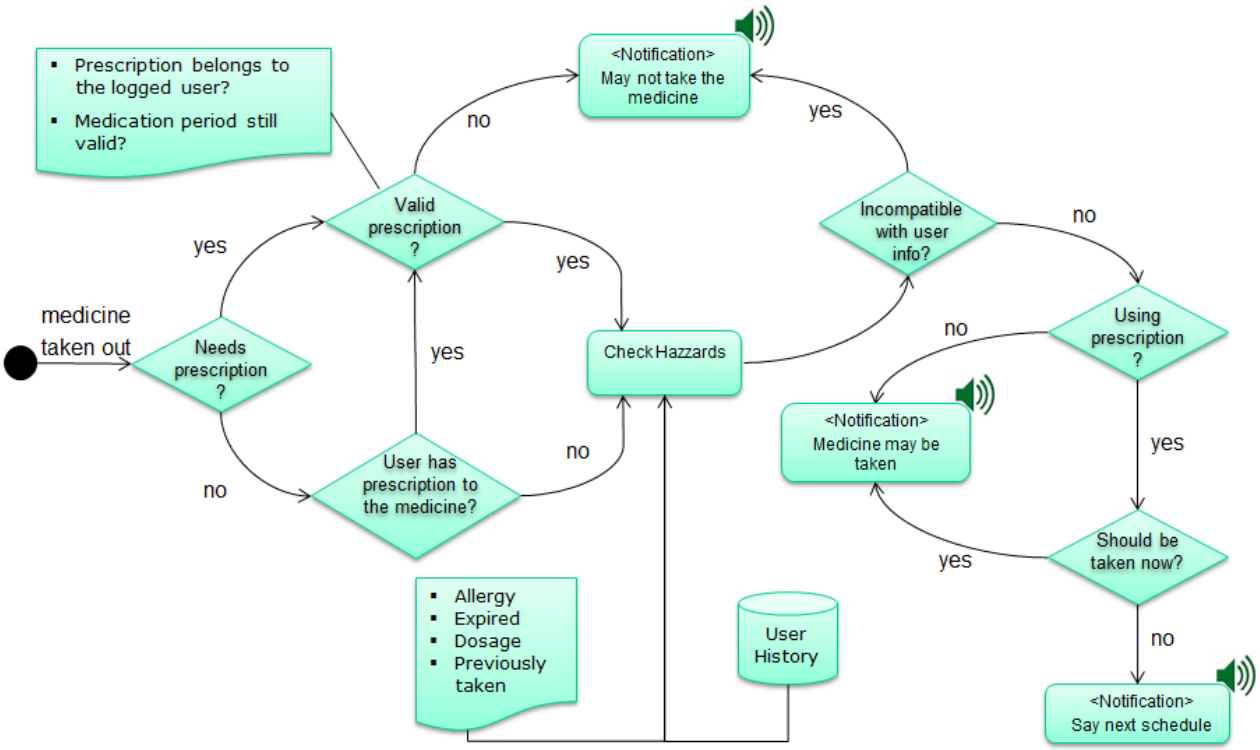

Figure 26: Main program logic

Furthermore, if a prescription medicine is to be taken, the prescription management of Figure 27 must be used. It is possible to see the four cases are considered when the user misses the right time of a schedule. A detailed explanation can also be found in the prototype chapter. But, as a basic description, if there is any tardiness, one of the four cases is used and the next schedule is calculated. If there was no delay, or the time for the medicine to be taken has been reached, the user is notified to take it.

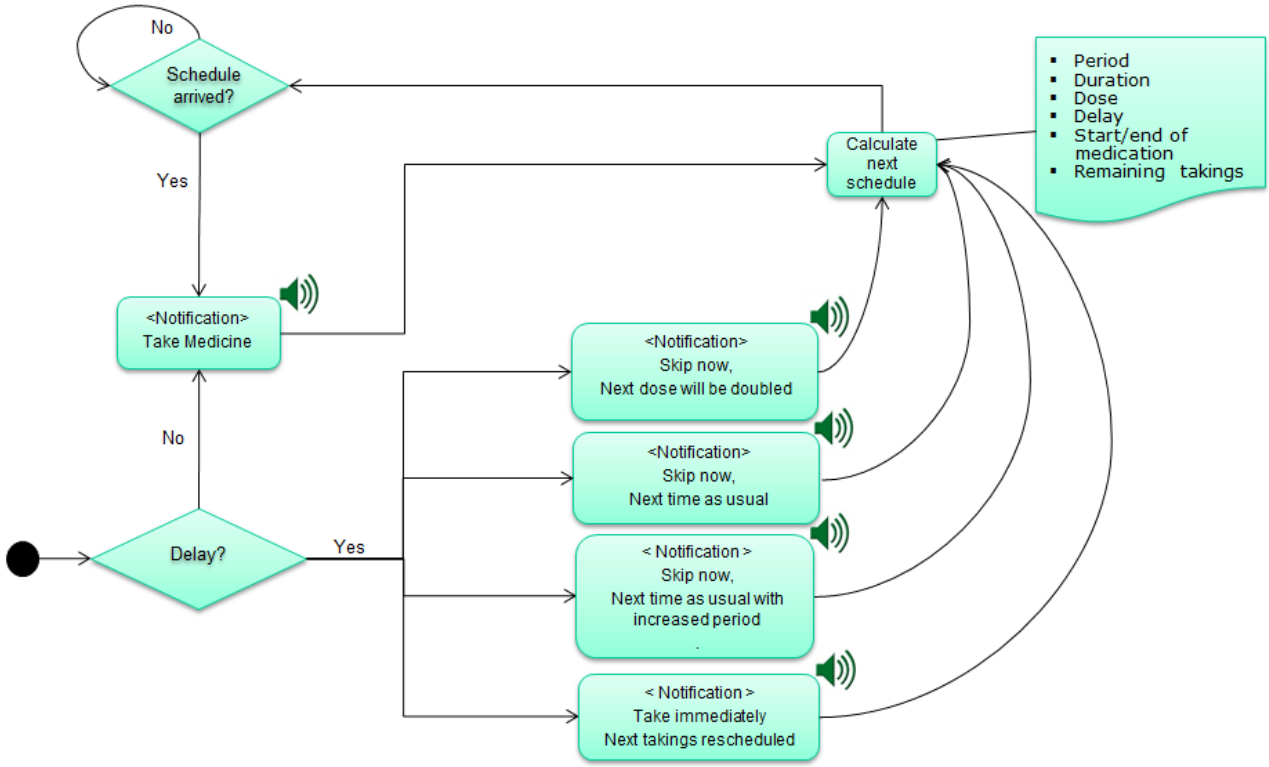

Figure 27: Program logic for the prescription

5.4.5 Web service

To access the medicine cabinet database from the web, it is necessary to provide a standardized way to read such data. Standardized way here means that it should be available to most of web applications: desktop browsers, mobile applications, smart TVs and etc. The most appropriate way to achieve such interoperability is a web service. As cited on W3C, there is a clear definition for it [42]: *"a web service is a software system designed to support interoperable machine-to-machine interaction over a network"*.

With a web service two different systems may not only communicate to each other, but also use functions from one another, regardless of the technologies used to implement them [43]. For the intelligent medicine cabinet, PHP (Hypertext Preprocessor) and JSON (Java Script Object Notation) are used to compose a web service, as it can be seen on Figure 28.

PHP stands for "PHP: Hypertext Preprocessor", a recursive acronym, and it is a server-side scripting language designed for web development, in other words, its processing is made on the server side and only the output is sent to the client. To format the output, JSON was chosen for this project. Its acronym stands for "JavaScript Object Notation and it is a light-

weight data interchange format if compared to other formats such as XML (Extensible Markup Language).

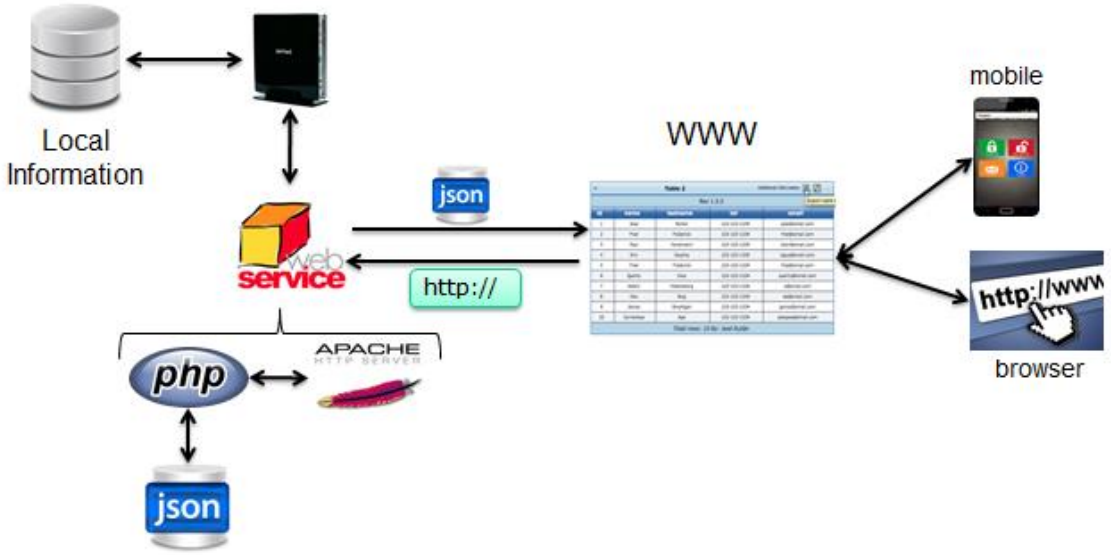

Figure 28: Medicine Cabinet web service

As can be seen, the workload has been removed from the client's side by using PHP and JSON, as the latter output is merely a text format that is completely language independent and easy for machines to parse and generate. Another important feature for which these two technologies were chosen is their availability. To use PHP it is only necessary to make HTTP requests, which is the most used protocol of the internet; and the JSON has been implemented in many programming languages and platforms [44].

To get information from the system data, PHP has built-in interface to access MySQL database. It makes the operations on the database and outputs the results in the JSON format. The intelligent medicine cabinet works as follows: 1) the server receives an "http" request from the web; 2) this requests is processed by an "Apache" server; 3) a PHP script processes the logic of this request and reads or writes to the database; 4) the PHP script writes the result in the JSON format which is then sent by the "Apache" server to the application which requested the operation.

Another advantage of using a web service with JSON is that its notation is human readable, that is, easy to be understood by any application or browsers which may use the web service. In a summarized version, the webs service operates receiving "http" inputs from an application and generating JSON outputs to them.

5.5 Chapter 5 highlights

In Chapter 4, the concept of the intelligent medicine cabinet showed how the problem of medication non-adherence could be solved. After the definition of the uses cases and the architecture of the system in this chapter, the first step towards the realization of the intelligent medicine cabinet was made. The use cases were important to show the possibilities for the user's interaction, while the architecture defined boundaries for the system development.

Once the definition of the limits of the system were made, it is now possible to develop a prototype based on this information, in other words, having the system functions specified is not sufficient, it is necessary to know how they can be realized by the use of the cited technologies. This chapter was fundamental to understand the intelligent medicine cabinet model, but the next one will be essential to see how it actually works.

Chapter 6- Prototype

On this chapter the intelligent medicine cabinet is explored into details, showing how each part of it was integrated to compose a prototype. At first, a brief description of how it looks like is made, but later a diagram with all of its software and hardware components is explained, showing the function of each element and how they communicate to each other.

Comparing the diagram with its prototype, it will be possible to see a difference between what the final user sees and what is behind the implementation of the solution. To support the understanding of how the prototype works, two scenarios are shown. Later, all the prototype functions are categorized in those related to each entity. Further details about how each element was implemented and extra functions may found from Appendix A to G on the end of this document.

6.1 Prototype Overview

Figure 29 shows the intelligent medicine cabinet prototype. In front of it only the LEDs are visible to provide visual signals to the user. All the other pieces of hardware are placed on its back, so that the user does not have to worry about how it was made. All the user has to do is to use their card to log in the system and start using it as a conventional medicine cabinet, as all the management is made automatically. It can, however, be locked using a manual lock, so that the management of whom uses it can still be made in case of power shortage.

Figure 29: Intelligent medicine cabinet prototype

In general terms, the intelligent medicine cabinet may be divided into two main parts which are depicted on Figure 30, which are the user's and system domain. The first one refers to the abstract or physical elements that can be accessed directly by the user. On such list of elements, there are the medicine box, user's card, prescription, cabinet door, light indications, instructions in the form of speech, and any device/software that can access data from the web service. The second one, however, is composed of the hardware and software elements that let the intelligent medicine cabinet manage the entities relationships and the user's interactions.

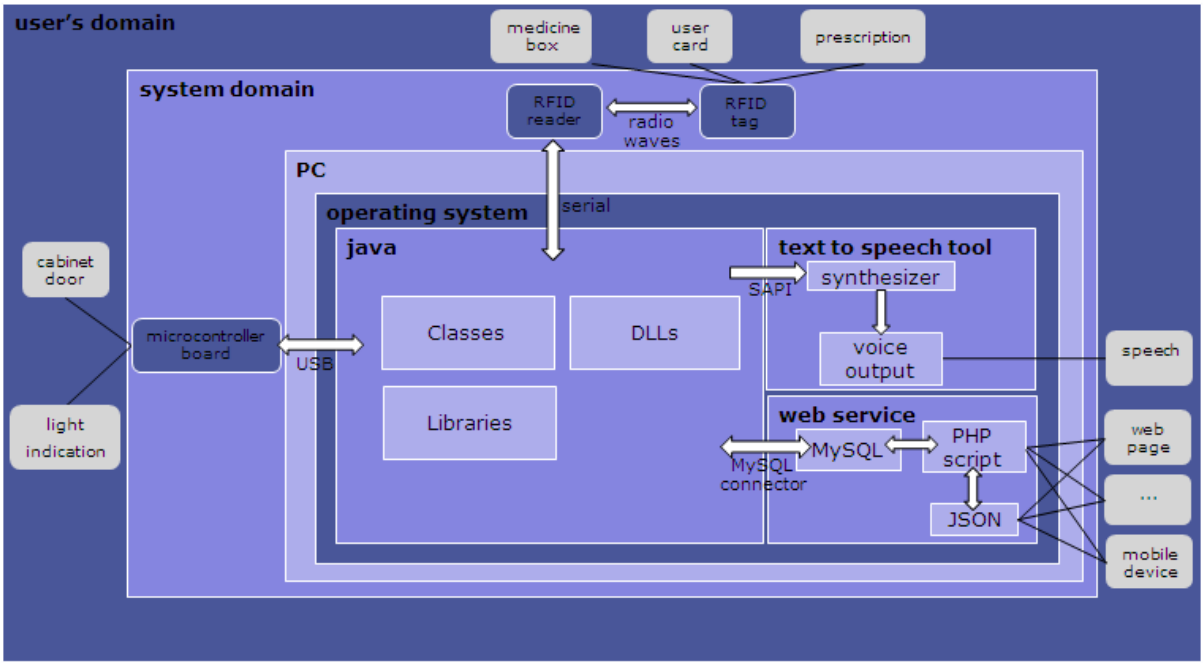

Figure 30: Prototype diagram

A PC is used to execute the source code, the web service and the database. The entities are detected wirelessly using the RFID technology and to store their information, RFID passive tags of the type ISO15693, with a frequency of 13.56 MHz, were used. This type of tags may also be read using NFC applications on a mobile device.

To detect the tags, a command is sent from the computer approximately every 250ms to the RFID reader to get the IDs of the tags in range. If a tag is added to the reading range, another command is sent to get its data, completing a period of about 300ms for the whole process, otherwise only the ID of the tags are read. If a tag is removed, the data captured when it was added is used. To know which tags were added or removed, the system keeps a

temporary list of the tags in range of each reading, so that, on the following reading, there is a list to make comparisons. That way, the system detects when prescriptions, users' cards and medicines are added to or removed from the cabinet and takes a decision upon that.

By combining the entities information, the system instructs the user depending on the situation (e.g. if an expired medicine is taken out, the user is warned about it). The program logic also compares the new information to those stored in a MySQL database, that way, as an example, it is possible to know if the user has previously taken an incompatible medicine with another one that is being taken out and a warning is generated. The programming language Java was used to implement the main program logic and it is composed of classes, supportive libraries and dynamic-link libraries (dlls). The last two are important to use APIs related to the speech synthesizer, serial communication [45], and database connector.

The main program has an exclusive class to access the system database using a MySQL connector. The synthesizer, which belongs to a text to speech tool, may be accessed via the SAPI. As can be seen on the figure, the system was modularized in such a way that the text to speech tool may be replaced by another one (e.g. that improves the naturalness of the speech) as much as it has compliance with SAPI.

A microcontroller of the PIC family was integrated on a board to either get input from a contact sensor or drive LEDs on and off as outputs. The contact sensor detects if the door is open and, by checking who is logged in the system (by using the user's cards information), grant access to the cabinet or not. The LEDs combine red, green and white colors and different blinking rates to indicate, in a categorized manner, when the user should pay more attention or should not take a medicine, for example. The language C was used to program the microcontroller.

Furthermore, there is the web service composed of the MySQL database and PHP and JSON modules. The database is the common interface between the main program and the web service because it is where all the cabinet usage information is concentrated. Figure 31 shows that the web service may be used by a web page, mobile application and any other device or software that may send HTTP request to it and read JSON responses.

As an example of a possible scenario, if someone, without a valid user card opens the cabinet door, this action is detected by the contact sensor and a signal is sent to the computer via the microcontroller. The computer, then, outputs a voice message and sends a command for the microcontroller to blink red LEDs, showing that this action is not allowed. The same would happen if a child, even with a valid card, tried to use it without an adult logged in at the

same time. On the other hand, if the user has a valid card, as depicted on Figure 31(a), the prototype detects it is in range and blinks green LEDs to show that access has been granted. As soon as the user opens the cabinet, this action is also detected and a welcome message is spoken along with green LEDs blinking Figure 31(b).

(a) (b)

Figure 31: Prototype used with a valid card

6.2 Prototype Functions

This section lists all the functionalities of the Intelligent Medicine Cabinet which will support the user in taking medicine and that were described into more details on the previous section. When the PC is turned on, the Intelligent Medicine Cabinet program is started. At the very beginning it checks for medicines, user's cards or prescriptions that were previously used on the system and loads them to the system memory, so that the user may have a smooth medication process after the start-up, without latency.

Just to remind, the medicine cabinet manages information about these three entities:

- User card – on this card the user's information is stored and the cabinet may be used.
- Medicine package – inside the medicine package there is an RFID tag which contains information about it.
- Prescription – behind the prescription there is an RFID tag attached to it. It has information on how the medication of each medicine on its list should be done.

Using information provided from these entities and combining them with information retrieved from the database, the medicine cabinet has many functions which are described on the next sections.

6.2.1 User

The following items list, in an atomic fashion, represents the functionalities of the intelligent medicine cabinet which are related only to the user:

- User may log in and out using their card
- If no user is logged in or a child is logged in, but there is no user logged in, the medicine cabinet warns the person that it shouldn't be used. The warning is first in the form of speech output when the door of the cabinet is open. But if the person insists and takes out a medicine, speech output is used combined with the sound of a siren.
- The medicine cabinet speaks the user's language. If the user's language is not supported, it uses English as default.
- Many users may be logged in the system at the same time, but only one user is active, so that the medicine cabinet knows which user is performing the actions.
- When a user logs out, the user who had logged in just before the current one becomes active.
- A user may have only one prescription at a time
- When the user logs in, it is checked if there were medicines taken in a period of 24 hours or that didn't have their effect already over. That way, the maximum daily dosage for a medicine is consistent and they user is no allowed to take incompatible medicines with those that didn't have their effect over.

6.2.2 Medicine

The following items list, in an atomic fashion, the functionalities of the intelligent medicine cabinet that are related to the interaction of user and medicine information, especially when a medicine is taken out of the cabinet, it is checked:

- if the medicine is expired

- if the medicine is almost empty
- if the user has allergy to the medicine
- if the user has incompatibility with the active ingredient group of the medicine
- if the user has incompatibility with the active ingredients of the medicine
- if the user has a prescription to the medicine. In case there is a prescription, the medicine is taken under a medication defined on the corresponding prescription
- if the maximum daily doses for the medicine has been reached
- if the medicine was recently announced as newly dangerous
- if the medicine has incompatibility with another medicine that was previously taken by the user

Furthermore, the following scenarios are taken into consideration:

- If the medicine needs a prescription to be taken, the user may only take the medicine if he or she has a valid prescription to the corresponding medicine
- If the user is using a prescription with medicines that are incompatible with the medicine taken out, the user is informed that he or she shouldn't take the medicine
- If the medicine is taken out for a long period of time, it is considered as removed from the cabinet
- If the medicine is placed into the cabinet again after a short period of time and there was no incompatibility with it, it is considered as taken

6.2.3 Prescription

The following items list, in an atomic fashion, the functionalities of the intelligent medicine cabinet that are related to the interaction of user and prescription. This interaction happens when a medicine under medication is used:

- When the user start a medication for a medicine, the next schedules and dosages are calculated
- If a user misses a schedule or is delayed, the medicine cabinet checks how the next schedule for that medication should be calculated. The four possible cases are: the user may take the medicine immediately and the next schedules are rescheduled based on the current time that the medicine was taken; the user should ignore the current schedule and take normal dosage on the next normal one, but the total

duration of the medication is increased; the user should ignore the current schedule and take double dosage on the next normal schedule; the user should take the medicine immediately and continue normally with the following schedules.

- When all the medicines of a prescription have their medications finished, the prescription is considered as not active anymore and may not be used. It may only be used, if the prescription is the type that allows repetition, that is, if the medicines on its list are taken for a period of time and there is a pause before they are taken again.
- A prescription is considered as active since at least one medication for one of its medicines is started.
- A prescription is considered as not active if the medication for its medicines hasn't started or are already finished
- When a user takes a medicine and it is not the right for it to be taken, the user is informed about the right time that it should be taken
- When the time for a medicine to be taken is reached, the user is informed with reminders that are generated periodically every 5 minutes
- When a medicine is taken, the Intelligent Medicine Cabinet calculates the next schedule, if there is some, and already tells the user about the next time that the medicine should be taken
- If a prescription is taken out of the cabinet for a certain period, it is considered as removed and its medications are considered as finished. Only if the prescription may be repeated (reused), it could be valid again.
- When the medicine cabinet is turned off and by that time it had an active prescription, the next time it is turned on again, the medications for the prescription are continued. If there were schedules to be taken during that time that the medicine cabinet was turned off, they are considered as taken.

6.2.4 Error Correction

If a user's card, prescription or medicine package is on an unstable region or position and the medicine cabinet may not read properly the information of the tag, it detects this behavior (e.g. consecutive "appearance" and "disappearance" of tags) and informs the user to

reposition it, so that after seconds, when stabilization is detected, the cabinet may be used normally.

6.3 Chapter 6 highlights

On this chapter an overview of system prototype was presented, showing how it looks like, and later each part of it was described into details. The components had their importance for the operation of the intelligent medicine described, such as: the PC, to which all components are connect and managed; a RFID reader, used to detect the entities automatically; a PIC microcontroller placed on a board, used for the visual signals and detect if the cabinet door is open; role of some libraries used to provide the functionalities discussed on the previous chapter; and how the result from the entities data abstraction was organized and stored on the prototype.

This chapter concludes the development of the intelligent medicine cabinet, which consisted of four phases. At first, a conceptual solution for the medication non-adherence was proposed. Then, on the definition phase, a system was modeled. On a third stage (i.e. design), an architecture was presented along with its components. And now, in the form of a prototype, the intelligent medicine cabinet was finally implemented. The results achieved so far can now be used to compare this master thesis with the state of art, which is the subject of the next chapter.

Chapter 7- Results

To determine if this master thesis has achieve positive results and that it accomplished what was proposed in the beginning of this document, the following things must be taken into consideration: the two tables from Section 3.10 which summarizes what has been proposed on the state of art; the specific and general objectives of this thesis; and the prototype requirements.

Those tables are presented here again, but they were combined into only one with all the issues and only the results for this master thesis are presented. A "+" signal means the issue had a positive solution. On the other hand, a "*" signal also represents a positive solution, but some observations are necessary to determine to which extent they can be used.

The creation of the two tables on section 3.10 let this master thesis accomplish its first two specific objectives. The first (i.e. *investigate and evaluate the strong and weak features of existing systems*) was accomplished by comparing all the related works; and the second one (i.e. *propose a concept based on the state of art*) was accomplished on section 4.2 where the concept of the intelligent medicine cabinet was proposed in such a way that the issues from the two tables could be approached. The completeness of the table issues will be discussed below.

This master thesis solved the first five issues from Table 7 by the use of RFID technology combined with a data abstraction that let the entities be separated from each other, enriching the importance of their relationships. The *"intake detection"*, however, is prone to failure if the patient only takes out a medicine, without actually taking it. But with that, the sixth specific objective was achieved, as the patient may use the cabinet as if it was a conventional one (i.e. *"natural interaction"* from Table 7). Furthermore, by the use of a multi-process (i.e. multi-thread) solution, it was possible to compute properly these relationships and provide a *"fast response"* to the user, covering the fifth specific objective of this thesis in such a way that the system may work even with the users' erroneous actions.

Table 7: State of art issues

Technological related issues	Results	User related issues	Results
Automatic entity detection	+	Privacy	*
Automatic intake detection	*	Usage by children	+
Many items at a time	+	Only allowed users	*
Unique entity detection	+	User incompatibility	+
Many items at a time	+	Medicine incompatibility	+
Fast response	+	Package status	+
Multi-device	*	Effective reminder	*
Speech capability	+	Prescription complexity	+
Message categorization	+	Schedule tardiness	+
Recall management	+	Medication flexibility	+
Medication status recovery	*	Medication dosage	+
Multiple users	+	User engagement	+
Natural interaction	+	User's language	*
Understanding of technology purpose	+	Interaction with caregiver	+
Updated data	+	Usage history incompatibilities	+
Information sharing	+		

The proposed system not only provides *"message categorization"* with red, white and green LEDs to represent types of message, but it also instructs the user by speaking their language with its *"speech capability"* that uses English as default in case their language is not supported. By the use of the web service with technologies that are wide spread, it was possible to implement "*recall management*" functionality, add "*updated data*" to the system, and provides *"information sharing"* of the usage history of the cabinet so that a caregiver

may monitor the patient's progress and even send them messages, covering the issue *"interaction with caregiver"* and accomplishing the third and fourth specific objectives of this thesis. The level of *"privacy"*, however, depends on where and to whom the user allows their information to be accessed. In addition, the developed cabinet may be classified as *"multi-device"*, since the web service offers part of it via web.

The concept also covered many incompatibilities that may exist during the medication process, such as: *"medicine incompatibility"*, on which it is checked if a medicine is not incompatible with another that the user intends to take (e.g. with a prescription one, that has priority); *"user incompatibility"*, on which allergies and incompatibilities with active ingredients and group of medicines are checked; *"usage history incompatibilities"*, so that the previous two issues are covered, but taking in account past actions (e.g. incompatibility with a medicine taken hours before); *"package status"*, so that the user is warned not to take expired medicines or is notified of a package that contains less than five pills; and *"medication dosage"*, so that patients are warned not to take more pills than the recommended daily dosage or for their age group.

By having an active user, *"multiple users"* may be logged in on the medicine cabinet at the same time and the issue *"usage by children"* is also covered, since, for a child to use the cabinet, it is necessary that an adult be also logged in. If the child is alone or a person without a valid card tries to use the cabinet, a warning message along with a siren sound is triggered. That way, *"only allowed users"* may use the cabinet. To prevent the cabinet from being unavailable in case of power shortage, it does not have an electronic lock, so, even though the intake is not computed by an invalid ID usage, a medicine may be taken; and, if there was any intake schedule for the period of time without power, it is not lost, being considered as taken when power is available again. It is considered that the patient took manual control of the situation, bearing in mind that the person is the most interested on their medication. This is how *"medication status recovery"* can be achieved.

As mentioned above, the user needs a login to use the cabinet. But that is not necessary for the user to get reminders about their schedules, in other words, the concept generates reminders even though the patient is not in its vicinity. Besides of that, *"effective reminders"* are generated for a period of time until the medicine is taken. The user, however, may decide when to take a medication (e.g. when to start it) or when to stop it (e.g. due to bad side effects). This *"medication flexibility"* was gained with the concept of two different types of prescriptions. The management of what should be done in case of *"schedule tardiness"* was

also investigated and four possible solutions were implemented. With all these attributes, the *"prescription complexity"* issue was covered in such a way that the patient does not have to take care of these parameters alone, having now a proper guidance.

Although most of the prototype requirements were already present on the paragraphs above, some of them need a description more focused on the technological aspects. It was required that the medicine cabinet should have a professional speech synthesizer and a modular system was designed, in such a way that more languages can be easily installed to consider the user's language, without the need of changing program code. Along with the voice outputs, it was required the use of useful visual signals, which were provided by the microcontroller PIC18F4550 using USB interface [46]. This microcontroller also provided a way to detect if the door of the cabinet was open, covering the requirement in avoiding hazardous usage by children. The concept was extended so that it could be used by children only if there was another user logged in the system or if the user had a valid card to log in.

The system also provided a web service to fulfill the call-back function in case a medicine is announced as newly dangerous, and to provide statistics of history of usage of the medicine cabinet. The web service was later extended to allow the addition of information to the cabinet database. In terms of performance, by using a multi-thread programming, it was possible to have a dedicated process to read and detect the RFID tags, making the medicine cabinet faster and detecting all the changes made by the user, instead of waiting for the speech outputs to finish and then read the tags again.

With all the information mentioned above, the seventh specific objective (i.e. *the prototype proposed should serve as a basis to apply the concept of this master thesis aiming the medication adherence*) was accomplished. With the simplicity of the prototype, the user may have a clearer view about the concept, solving the *"understanding of technology purpose"* (from Table 7) issue. Furthermore, with a simple system that does not requires much workload from the patients, *"user engagement"* (from Table 7) is gained, since the user is not stressed with extra tasks and may have more time to focus on their medication.

Chapter 8- Conclusion and Future Work

The related work discussed on this master thesis showed how the medication process might become dangerous to the user, due to its many parameters and improper regimens. It proposed an intelligent medicine cabinet concept that uses RFID technology to identify automatically user's, medicine and prescription information. By processing the relationships among these entities, it manages user's allergies, medicine incompatibilities, medicine expiry date and remaining quantity of pills.

The advantage of using such concept is that users do not have to take care alone of the many parameters involved on the medication process, as they are instructed by the cabinet via a speech synthesizer and visual signals. All these features are an attempt to solve the problem of medication non-adherence. The instructions are used to tell if a medicine should be taken or not, the right time to take it and what should be done in case one schedule is missed. Furthermore, only valid users may use the cabinet, avoiding wrong usage by those who do not have a card to be identified or by children not supervised and it takes care of the maximum allowed dosage rate of medicines.

Another great functionality that was developed for the intelligent medicine cabinet is its ability in working offline, at least partially. By the use of a manual lock, the patients are not prevented from using the cabinet in case of a power shortage. The same principal applies for the prescriptions management, as, even though the reminders are not generated because of the power failure, the system tries to recover the medications status while the cabinet was turned off. This recovery mode serves as an example of such functionality by considering that during the power shortage, the user, who is interested on their health, took manual control of their schedules. That way, when the system is turned on, it considers that the schedules were taken manually. This feature is very important in Brazil, especially in cities where power failure is a constant problem.

Throughout the development of the intelligent medicine cabinet many challenges were faced. The first one was in finding a professional speech synthesizer with low cost. Most of the speech engines cost around 1000 € and some of them only provided temporary licenses.

After a lot of negotiation and testing engines APIs, it was found out that the cheapest solution was to buy a software that used most of the voices available on the market and integrate the voices myself. To achieve this, the voices needed to be compatible with Microsoft SAPI API.

Another challenge, I would say the most difficult one, was the RFID reader. As it was discussed in Section 5.4.2, the system had to be designed in such a way that it should be possible to have a dedicated process to read the RFID tags. With that, a lot of error correction was added to handle the errors induced not only by tags in a wrong position or unstable reading region, but also actions performed by the user, which are easily done with a conventional medicine cabinet. Another problem that was verified is the way that Windows enables serial communication: as there are many threads running in the operating system, there is no deterministic moment to have its data available. That way, many tests were made to find the best timing to read the tags, that is, the right frequency on which data should be requested from the RFID reader. Also, many possibilities of interaction were tested to get useful information about the user's interaction and to try to solve them.

The third greatest challenge was the development of the prescription concept. A lot of search and reading was necessary to evaluate what the most common actions are to be performed when a user forgets to take a medicine on the right time. I even consulted health professionals to check if my solution was reasonable and also learnt with them through their feedback. The temporal behavior of the prescriptions increased the level of difficultness in both calculating schedules and in rebuilding the user's medication, using information from the database, if they medicine cabinet is turned off, for example.

The last challenge refers to the fact that, as this master thesis was made in partnership with the "Institut für Automatisierungs- und Softwaretechnik", I had to follow the IAS-Quality model, which uses the V-model of Software Engineering development, that provides a way to keep traceability throughout the whole master thesis. Among the difficulties faced there, I could highlight: the language, as not everybody spoke English and, for the confection of the microcontroller board and operating system installation, German had to be used; and that IAS represented a very demanding "client", as the intelligent medicine cabinet is part of their showcase (i.e. the public was interested), which means that there was a lot of pressure to deliver something working correctly.

As future work, more languages could be included (e.g. Spanish, which is wide spoken in Latin America). Other technologies could be used to identify the user instead of only using an RFID card. A camera could be used to recognize the user or the medicine intakes more

precisely. By using the services of the web service, the medicine cabinet could provide its information to digital TVs, smartphones and other mobile devices.

With the web service, it could be also possible to get medicines information from the web automatically and insert it to cabinet database. The medicine cabinet could have its web service extended for more maintenance work, for example, to update the system source code or check the status of the medicine cabinet.

Another important step as future work is to test the intelligent medicine cabinet with real users of different age groups. With their feedback, errors could be found; certain features could be improved, so that unused features could be removed, and those still considered complex by them, could be simplified. With real users, we could verify the real effectiveness of the system, in other words, if it indeed reduces medication non-adherence.

Finally, to summarize all the work done, the intelligent medicine cabinet is a concept developed in an attempt to facilitate people's lives in taking medicines, in such a way that the system may be used as simple as using a conventional medicine cabinet, but increasing medication adherence. By using it, if a person says that he or she had to learn almost nothing to use it, than it is an indicative that the main goal of this thesis has been reached: be as close as possible to what people expect when taking medicines, but provide a smooth, easy, and guided medication process.

References

[1] REPORTER, D. M. *Parents sue after toddler son' pukes up blood and dies after taking recalled Children's Tylenol'*. 2012. Available: <http://www.dailymail.co.uk/news/article-2083218/Parents-suing-Johnson--Johnson-toddler-dies-taking-recalled-Childrens-Tylenol.html>. Accessed: 20th May 2013.

[2] JAMES, S. D. *5-Year-Old Colorado Girl Dies of Cough Medicine Overdose*. 2012. Available: <http://abcnews.go.com/Health/year-coloradogirl-dies-cough-medicine-overdose/story?id=16151167>. Accessed: 20th May 2013.

[3] SPAN, P. *A Dose of Confusion*. 2011. Available: <http://newoldage.blogs.nytimes.com/2011/06/15/a-dose-of-confusion/>. Accessed: 20th May 2013.

[4] GIRION, L.; GLOVER, S.; SMITH, D. *Drug deaths now outnumber traffic fatalities in U.S., data show*. 2011. Available: <http://articles.latimes.com/2011/sep/17/local/la-me-drugs-epidemic-20110918>. Accessed: 20th May 2013.

[5] KOCHANEK, K. D.; KIRMEYER, S. E.; MARTIN, J. A.; STROBINO, D. M.; GUYER, B. *Annual summary of vital statistics: 2009*. Pediatrics, Am Acad Pediatrics, v. 129, n. 2, p. 338–348, 2012.

[6] SUN, H.; FLORIO, V. D.; GUI, N.; BLONDIA, C. Promises and challenges of ambient assisted living systems. In: *Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on*. [S.l.: s.n.], 2009. p. 1201–1207.

[7] WHITE, C.; FANG, D.; KIM, E.-H.; LOBER, W.; KIM, Y. Improving healthcare quality through distributed diagnosis and home healthcare (d/sub 2/h/sub 2/). In: *Distributed Diagnosis and Home Healthcare, 2006. D2H2. 1st Transdisciplinary Conference on*. [S.l.: s.n.], 2006. p. 168–172.

[8] ZATOUT, Y. Using wireless technologies for healthcare monitoring at home: A survey. In: *e-Health Networking, Applications and Services (Healthcom), 2012 IEEE 14th International Conference on*. [S.l.: s.n.], 2012. p. 383–386.

[9] COOPMAN, T.; THEETAERT, W.; PREUVENEERS, D.; BERBERS, Y. A user-oriented and context-aware service orchestration framework for dynamic home automation systems. In: AUGUSTO, J.; CORCHADO, J.; NOVAIS, P.; ANALIDE, C. (Ed.). *Ambient Intelligence and Future Trends-International Symposium on Ambient Intelligence (ISAmI 2010)*. [S.l.]: Springer Berlin Heidelberg, 2010, (Advances in Intelligent and Soft Computing, v. 72). p. 63–70. ISBN 978-3-642-13267-4.

[10] CANTRILL, M.; STEPHEN, V. Computers in patient care: The promise and the challenge. *Queue*, ACM, New York, NY, USA, v. 8, n. 8, p. 20:20–20:27, Aug 2010. ISSN 1542-7730.

[11] MCHOME, S.; SACHDEVA, S.; BHALLA, S. A brief survey: Usability in healthcare. In: *Electronics and Information Engineering (ICEIE), 2010 International Conference On*. [S.l.: s.n.], 2010. v. 1, p. V1–463–V1–467.

[12] ISO. TS 16071: 2003: Ergonomics of human-system interaction--Guidance on accessibility for human-computer interfaces. *International Standards Organisation, Geneva, Switzerland*, 2003.

[13] KINSELLA, K.; HE, W. *An aging world: 2008*. In: . Washington, DC, USA: US Government Printing Office, 2009. v. 95, p. 7–18.

[14] NATIONS, U. *UN Population Division Policy Brief No. 20091*. Mar 2009. Available: <http://www.un.org/esa/population/publications/UNPD_policybriefs/UNPD_policy_brief1.pdf>. Accessed: 20th May 2013.

[15] ROBERTS, C. Radio frequency identification (RFID). *Computers & Security*, v. 25, n. 1, p. 18–26, 2006. ISSN 0167-4048.

[16] ISO. ISO/IEC 15693-2:2006 Identification cards -- Contactless integrated circuit cards -- Vicinity cards -- Part 2: Air interface and initialization. *International Standards Organisation, Geneva, Switzerland*, 2013.

[17] ISO. ISO/IEC 18000-3:2010 Information technology -- Radio frequency identification for item management -- Part 3: Parameters for air interface communications at 13,56 MHz. *International Standards Organisation, Geneva, Switzerland*, 2010.

[18] ISO. ISO/IEC 15426-2:2005 Information technology – Automatic identification and data capture techniques -- Bar code verifier conformance specification -- Part 2: Two-dimensional symbols. *International Standards Organisation, Geneva, Switzerland*, 2009.

[19] ISO. ISO/IEC 15416:2000 Information technology – Automatic identification and data capture techniques -- Bar code print quality test specification -- Linear symbols. *International Standards Organisation, Geneva, Switzerland*, 2008.

[20] ISO. ISO/IEC 14443-4:2008 Identification cards -- Contactless integrated circuit cards -- Proximity cards -- Part 4: Transmission protocol. *International Standards Organisation, Geneva, Switzerland*, 2008.

[21] COSKUN, V.; OK, K.; OZDENIZCI, B. *Near Field Communication (NFC): From Theory to Practice*. [S.l.]: John Wiley & Sons, 2011. ISBN 9781119966906.

[22] MICROSYSTEMS, S. *Java Speech API Programmer's Guide v 1.0*. [S.l.]: Sun Microsystems, Inc Business, 1998. Available: <http://java.coe.psu.ac.th/Extension/JavaSpeech1.0/jsapi-guide.pdf>. Accessed: 20th May 2013.

[23] OWNBY, R. L. Medication adherence and cognition medical, personal and economic factors influence level of adherence in older adults. *Geriatrics*, NIH Public Access, v. 61, n. 2, p. 30, 2006.

[24] IGLESIAS, R.; PARRA, J.; CRUCES, C.; SEGURA, N. G. de. Experiencing NFC-based touch for home healthcare. In: ACM, Corfu, Greece. *Proceedings of the 2nd international*

conference on pervasive technologies related to assistive environments. New York, NY, USA: ACM, 2009. (PETRA '09), p. 27:1–27:4. ISBN 978-1-60558-409-6.

[25] HO, L.; MOH, M.; WALKER, Z.; HAMADA, T.; SU, C.-F. A prototype on RFID and sensor networks for elder healthcare: progress report. In: ACM, Philadelphia, Pennsylvania, USA. *Proceedings of the 2005 ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis*. New York, NY, USA, 2005. (E-WIND '05), p. 70–75. ISBN 1-59593-026-4.

[26] SIEGEMUND, F.; FLORKEMEIER, C. Interaction in pervasive computing settings using bluetooth-enabled active tags and passive RFID technology together with mobile phones. In: *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on*. [S.l.: s.n.], 2003. p. 378–387.

[27] HASANUZZAMAN, F.; YANG, X.; TIAN, Y.; LIU, Q.; CAPEZUTI, E. Monitoring activity of taking medicine by incorporating RFID and video analysis. *Network Modeling Analysis in Health Informatics and Bioinformatics*, Springer Vienna, v. 2, n. 2, p. 61–70, 2013. ISSN 2192-6662.

[28] LOPEZ-NORES, M.; PAZOS-ARIAS, J.; GARCIA-DUQUE, J.; BLANCO-FERNANDEZ, Y. Monitoring medicine intake in the networked home: The iCabiNET solution. In: *Pervasive Computing Technologies for Healthcare, 2008. PervasiveHealth 2008. Second International Conference on*. [S.l.: s.n.], 2008. p. 116–117.

[29] CALABRETTO, J.-P.; WARREN, J.; DARZANOS, K.; FRY, B. Building common ground for communication between patients and community pharmacists with an internet medicine cabinet. In: System Sciences, 2002. HICSS. *Proceedings of the 35th Annual Hawaii International Conference on*. [S.l.: s.n.], 2002. p. 2087–2094.

[30] CALABRETTO, J.-P.; COUPER, D.; MULLEY, B.; NISSEN, M.; SIOW, S.; TUCK, J.; WARREN, J. Agent support for patients and community pharmacists. In: *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*. [S.l.: s.n.], 2002. p. 2003–2012.

[31] FERREIRA, F.; ALMEIDA, N.; ROSA, A.; OLIVEIRA, A.; TEIXEIRA, A.; PEREIRA, J. Multimodal and adaptable medication assistant for the elderly: A prototype for interaction and usability in smartphones. In: *Information Systems and Technologies (CISTI), 2013 8th Iberian Conference on*. [S.l.: s.n.], 2013. p. 1–6.

[32] TANG, L.; ZHOU, X.; YU, Z.; LIANG, Y.; ZHANG, D.; NI, H. Mhs: A multimedia system for improving medication adherence in elderly care. *Systems Journal, IEEE*, v. 5, n. 4, p. 506–517, Dec 2011. ISSN 1932-8184.

[33] TIWARI, P.; WARREN, J.; DAY, K.; MACDONALD, B.; JAYAWARDENA, C.; KUO, I. H.; IGIC, A.; DATTA, C. Feasibility study of a robotic medication assistant for the elderly. In: *Proceedings of the Twelfth Australasian User Interface Conference - Volume 117*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2011. (AUIC '11), p. 57–66. ISBN 978-1-920682-97-2.

[34] LEE, Y.; TULLIO, J.; NARASIMHAN, N.; KAUSHIK, P.; ENGELSMA, J.; BASAPUR, S. Investigating the potential of in-home devices for improving medication adherence. In: *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on*. [S.l.: s.n.], 2009. p. 1–8.

[35] FDA, U. U.S. *Food & Drug Administration Online Therapeutical Records*. Available: <http://www.fda.gov/default.htm>. Accessed: 20th May 2013.

[36] ANVISA. *Anvisa Therapeutical Records*. Available: <http://www.anvisa.gov.br/datavisa/fila_bula/index.asp>. Accessed: 20th May 2013.

[37] KATZUNG, B. G.; MASTERS, S. B.; TREVOR, A. J. *Basic & clinical pharmacology*. New York, NY, USA: Lange Medical Books/McGraw-Hill, 2004. (LANGE Basic Science). ISBN 0071764011.

[38] ZÜHLKE, D. *Useware-Engineering für technische Systeme*. [S.l.]: Springer Berlin, Heidelberg, New York, 2004.

[39] AHO, A. V.; SETHI, R.; ULLMAN, J. D. *Compilerbau*. [S.l.]: Addison-Wesley, 1988.

[40] SCHWEITZER, G. Mechatronics for the design of human-oriented machines. *Mechatronics, IEEE/ASME Transactions on*, v. 1, n. 2, p. 120–126, June 1996. ISSN 1083-4435.

[41] YAZDI, F.; VIERITZ, H.; JAZDI, N.; SCHILBERG, D.; GOEHNER, P.; JESCHKE, S. A concept for user-centered development of accessible user interfaces for industrial automation systems and web applications. In: STEPHANIDIS, C. (Ed.). *Universal Access in Human-Computer Interaction. Applications and Services*. [S.l.]: Springer Berlin Heidelberg, 2011, (Lecture Notes in Computer Science, v. 6768). p. 301–310. ISBN 978-3-642-21656-5.

[42] BOOTH, D.; HAAS, H.; MCCABE, F.; NEWCOMER, E.; CHAMPION, M.; FERRIS, C.; ORCHARD, D. Web services architecture, W3C Working Group note, 11 february 2004. *World Wide Web Consortium*, 2004. Article available from:: <http://www.w3.org/TR/ws-arch>.

[43] DAIGNEAU, R. *Service Design Patterns: fundamental design solutions for SOAP/WSDL and restful Web Services*. [S.l.]: Addison-Wesley, 2012.

[44] ORG, J. *ECMA-404 The JSON Data Interchange Standard*. Available: <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>.
Accessed: 20th May 2013.

[45] AXELSON, J. *Serial Port Complete Second Edition: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems*. [S.l.]: Lakeview Research, 2007.

[46] LUNT, B. D. *USB: The Universal Serial Bus*. [S.l.]: CreateSpace Independent Publishing Platform, 2012. (Operating System Design (Book 8)). ISBN 1468151983.

Appendix A- Prototype Hardware Components

Hardware Overview

Figure 32 shows the system hardware model with its interfaces. It contains four main hardware blocks: RFID reader, RFID tag, PC station, and Microcontroller board. The elements in blue are the components of each hardware block. Some of the components are hardware and others software. The system has three interfaces: radio waves (between the RFID reader and the tags), serial (between RFID reader and PC station), and USB (between PC station and Microcontroller board). The hardware and software modules are described in the upcoming sections.

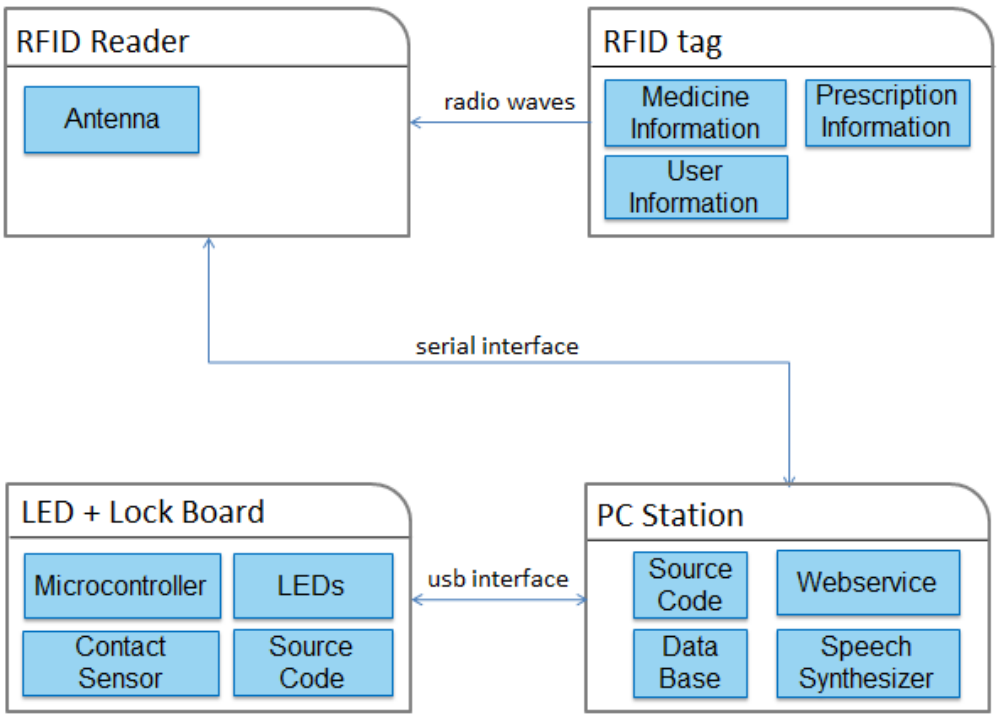

Figure 32: System hardware model

PC Station

The Zotac ZBOX SD-ID13 is the PC used to run the source code, the webserver and the database. Figure 33 shows the PC used in Germany for the prototype. Among its many features, the following ones are important for the use by the intelligent medicine cabinet: Ethernet connection; it is compact, measuring 188x188x44 mm; it has a minimum of 2 USB ports; and its Intel® Atom™ D525 (2 x 1.8 GHz) processor has enough speed to process all the communications requests and source codes of the medicine cabinet. Although the computer used in Brazil is not the same, it differs most from the Mini-PC with its size.

Figure 33: PC Station ZBOX SD-ID13

RFID Reader

For the intelligent medicine cabinet to read the transponders of the medicines, users' cards and prescriptions, an RFID reader is necessary. The reader "FEIG OBID i-scan HF Mid Range Reader ID ISC.MR102-A" was chosen to perform this task. Figure 34 shows the reader.

Figure 34: RFID Reader

It needs a supply voltage of 12-24 V DC and it uses an RS232 interface to communicate to the PC. It can read RFID tags of the type ISO15693, which are used in this project. It has an anti-collision algorithm, which enables the reading of multiple transponders at a time. That is exactly what the medicine cabinet needs, as many medicine, user or prescriptions tags may be present in the cabinet at the same time.

It has three modes of operation: Scan, FEIG ISO HOST and Notification. Using the FEIG ISO HOST mode it is possible to send commands to the RFID reader, which enables it to read only the transponders IDs at a time or to address one tag, by using its ID, and request its specific data. That way, the reader may detect if a new transponder was added to or removed from its range, reading the IDs and requesting its data, if a new one is detected nearby.

With a compact size of 145x85x31 mm, its reading area is not enough for a medicine cabinet with many tags. Therefore, an antenna must be used to increase the reader range. For this project the antenna of Type "ID ISC.ANT340/240" was employed to not only increase the reading range up to 30 cm, but also increase the reading area up to 337x237 mm, compared to the 145x85 mm area of the reader. Figure 35 shows such antenna.

Figure 35: RFID Antenna

The RFID tags (transponders) used to identify and store data of the medicines, users and prescriptions work on a frequency of 13.56 MHz and are compliant to ISO15693. The medicine and prescription tags look like a normal piece of paper (Figure 36) and may be easily inserted into the medicine packages or attached to the prescription papers.

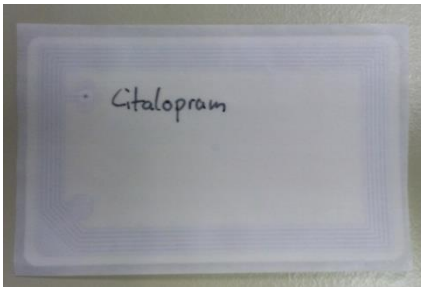

Figure 36: Medicine transponder

The user tags come in the form of a plastic card and are water proof. That way, even children may use them without the worry that it would be easily damaged.

Figure 37: User transponder

Microcontroller PIC18f4550

To drive the LEDs on and off and to detect if the medicine cabinet is open, not only an interface with input/output signals is necessary, but also an interface to get and send this signals from the PC station. The PIC18f4550 was chosen for this purpose. It has the following features: full speed USB 2.0 (12Mbit/s) interface; 1K byte dual port RAM + 1K byte GP RAM; full speed transceiver; 16 endpoints (IN/OUT); streaming port; internal pull up resistors (D+/D-); 48 MHz performance (12 MIPS); and pin-to-pin compatible with PIC16C7X5.

With its large amount of RAM memory for buffering and its full speed USB interface, it is ideal for the embedded control with low consumption and monitoring application that is required by the medicine cabinet. Such hardware demand is necessary because of the periodic connection with the PC via USB for data exchange.

Furthermore, this powerful microcontroller may enable future addition of input/output signals to the system functionality, for example, a sensor to detect the presence of a user or an automatic door system for the cabinet. The main reason for it to be chosen was its USB module (Figure 38). The use of external pull-up resistors and external supply voltage is optional, as it already has internal pull-up resistors for the D- and D+ USB pins and the microcontroller may be "bus-powered", in other words, it draws its power from the USB host (from the PC). That way, no power regulation is required and the connection to the PC is straight forward (no need of serial-converters). The whole work must, then, be done in the software.

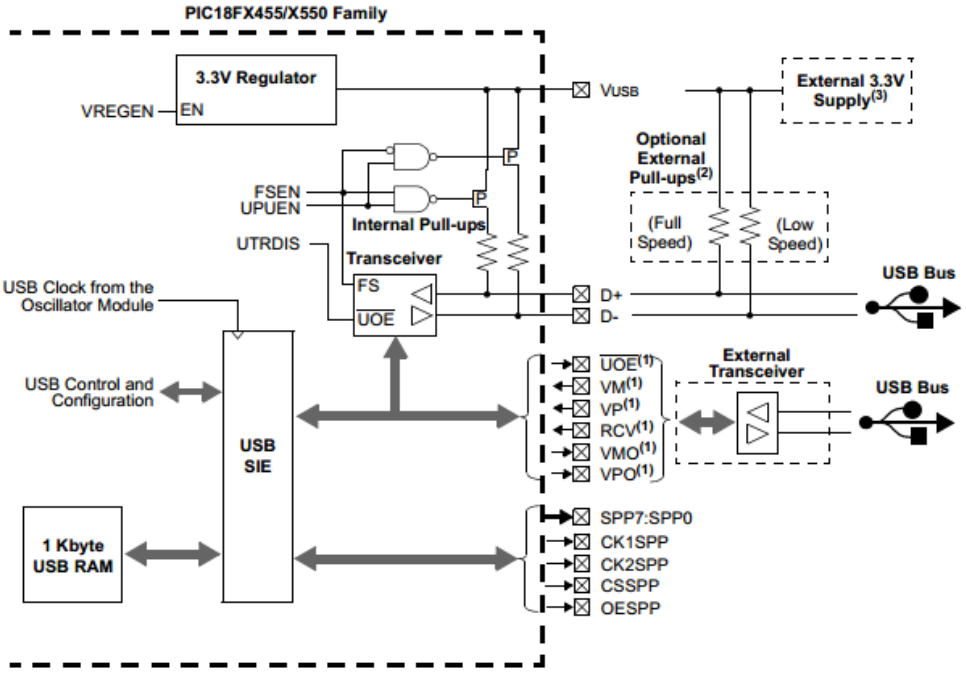

Figure 38: PIC18f4550 USB interface

Microcontroller Board

Figure 39 shows the Microcontroller board. It uses the PIC18f4550 to control the signals for the LEDs and the contact sensor. The LEDs have 3 colors: red, green and white. Their combination is used to classify warnings given to the user, as for example: if a medicine should not be taken, if user should pay attention to an event and etc. These combinations are bundled in commands sent from the PC to the microcontroller via USB interface. The contact

sensor stays attached to the cabinet door. If it is open or closed, a signal is sent to the microcontroller, which interprets it and sends a USB message to the PC.

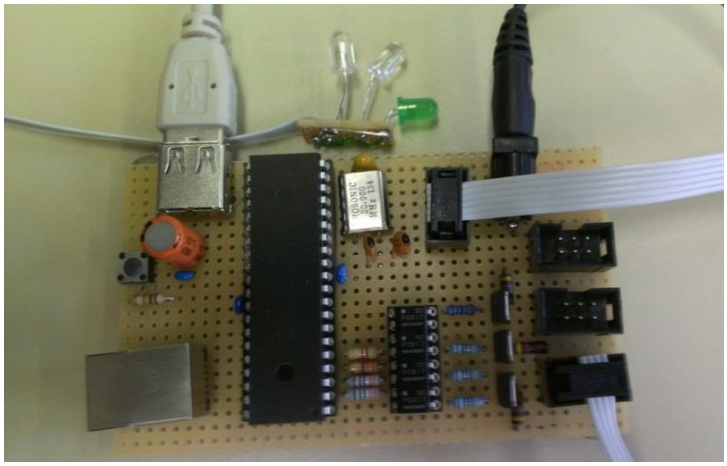

Figure 39: Microcontroller board

Figure 40 shows the schematic for the USB interface of the board, that is, only the components necessary to enable the USB communication with the microcontroller. Figure 41 shows the schematic to output the LEDs signals and get inputs from the contact sensor. As the microcontroller is already “bus-powered”, an external power supply is necessary for the LEDs. Figure 42 shows the size of the board compared to a coin of 1 cent of Euro.

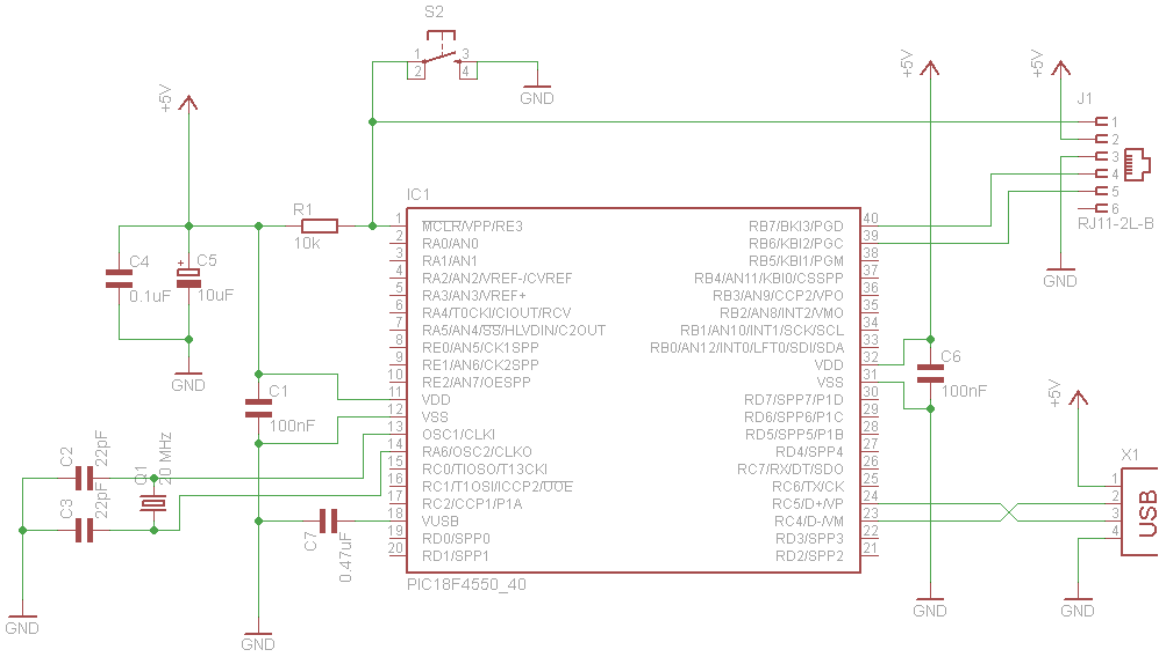

Figure 40: Board USB interface schematic

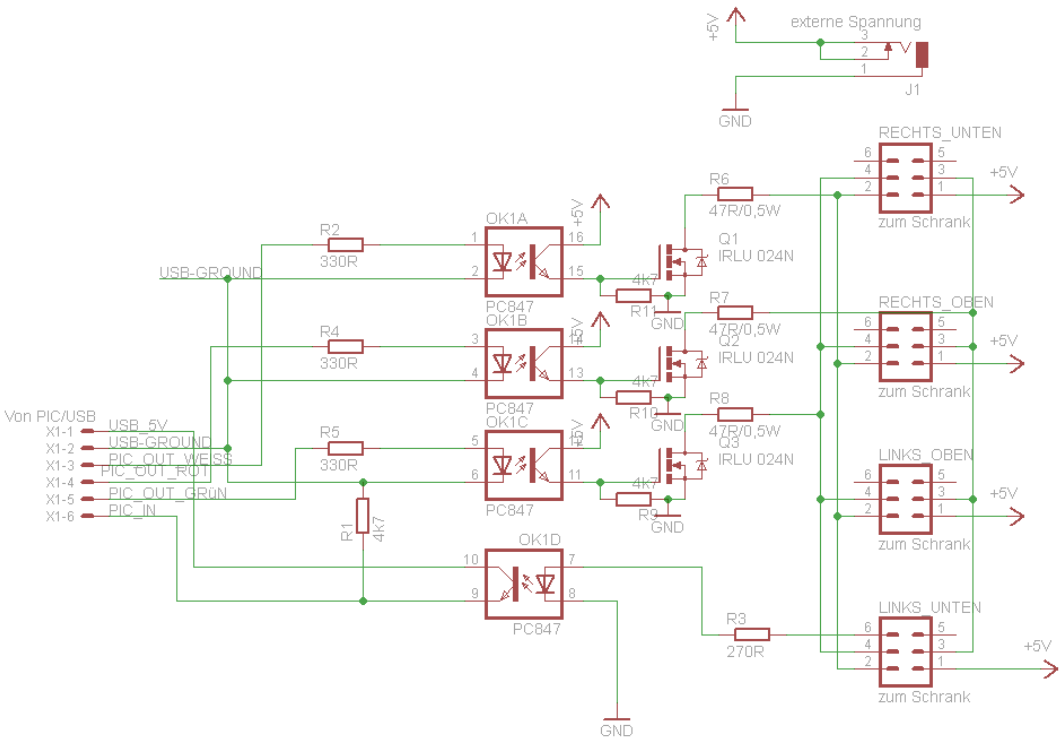

Figure 41: Board LEDs and contact sensor interface schematic

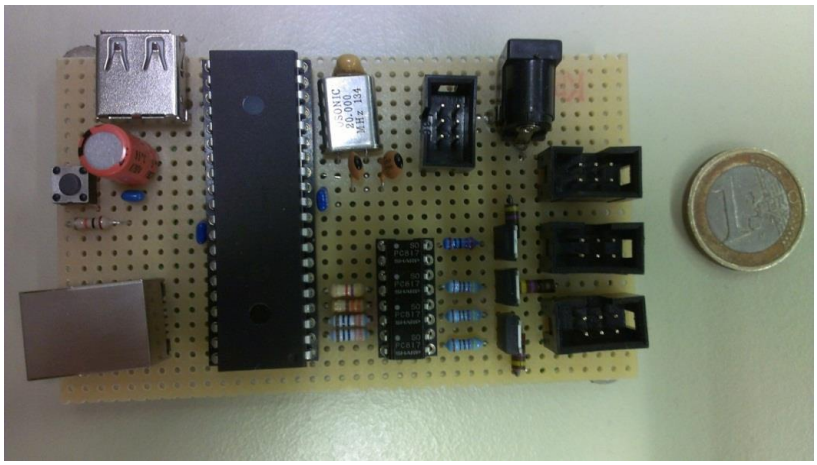

Figure 42: Microcontroller board sizing

Appendix B- Software Component: PIC Source Code

PIC18f4550 Source Code

For the USB communication of the microcontroller to work, the "Microchip USB Framework for PIC18, PIC24 & PIC32" was used. Among the many possibilities of the framework, the "USB Communications Device Class (USB CDC)" was used, so that the PC may interact with the microcontroller as if it was connected to a serial port. That means that there is no need of writing a USB driver.

If a USB driver was to be developed, the medicine cabinet would need a vendor ID (which may have a high cost) or use the Microchip ID. To avoid this, the USB CDC allows data to be sent and read from a simple virtual COM Port. Table 8 shows some of the most important functions of this framework.

Table 8: Microchip USB CDC

Function	Functionality
void USBCB_SOF_Handler(void)	The USB host sends out a SOF (start of frame) packet to full-speed devices every 1 ms. This interrupt may be useful for isochronous pipes. The counters for the blinking of the LEDs and duration that the LEDs must be on are update here.
void UserInit(void)	In this function all the user's application initializations should be performed. For the medicine cabinet, the LEDs and the contact sensor ports are initiated here.
BOOL USER_USB_CALLBACK_EVENT_HANDLER(USB_EVENT event, void *pdata, WORD size)	This function is called from the USB stack to notify a user application that a USB event occurred. This callback is in a interrupt context when the USB_INTERRUPT option is selected. For the medicine cabinet, this is exactly where the USBCB_SOF_Handler is called every 1 ms.
void user(void)	This function is reserved for the user's code itself. Here the user's logic must be done, that is, sending to and receiving data from the USB, as well as its processing.

The function “void user(void)”, as said above, is in charge of reading/writing to the USB. The best way to handle the communication is by the use of interruptions. By using such approach, it is not necessary to be polling the bus to check if a message has arrived or there is something to be transmitted. To do so, in the main loop the following step must be performed before trying to call “void user(void)”; check if there is any signal present on the bus with USB_BUS_SENSE (if not defined as 1) and if the “USBGetDeviceState()” is not in the DETACHED_STATE. Detached is the state in which the device is not attached to the bus. If the device is not attached, “USBDeviceAttach()” should be called to configure the device in the ATTACHED_STATE.

Once the device is attached, two things must be performed before trying to send or receive data from the bus: check if the USB is in the CONFIGURE_STATE, that is, if the device has been fully enumerated and is operating on the bus. If so, the device is now allowed to execute its application tasks (send and receive date); and check with USBSuspendControl if the bus is not suspended for any reason.

After that, it is possible to read/write data to the USB. “USBUSARTIsTxTrfReady()” is used to detect if the bus is ready to send or receive data; “getsUSBUSART(Buffer,size)” is used to read information from the bus; “putUSBUSART(Buffer,numBytesToSend)” is used to transmit data; and “CDCTxService()” is used to actually perform the reading and writing operations defined by the last two functions.

Appendix C- Software Component: Java Source Code

Java Libraries

For the medicine cabinet main code communicate with the microcontroller, generate reminders for the medication schedules, play the siren sound, and access its database, some libraries were necessary to provide such functionalities. They are listed below.

RXTX Serial

RXTX is a library that uses Java Natural Interface (JNI) to provide serial and parallel communication for the Java Development Toolkit (JDK). Besides being in accordance with the GNU LGPL license, it is only based on the specification for Sun's Java Communications API, though many of the class descriptions are the same, such package did not use it, since gnu.io is used instead. A certain amount of compatibility is intended with such API, though their project should be considered as a fork and therefore compatible in spirit, but not in implementation with the core library of Sun. This library is intended to enable the serial communication between the PC and the RFID reader and the PC and the Microcontroller.

Quartz Scheduler

The Quartz Scheduler is used to manage the temporal behavior of the medicine cabinet, in other words, to trigger its reminders: when a user has to take a medicine, when a medicine effect is over, when a medicine expires and so on. Quartz is a small java library that contains all the core of Quartz functionality. The main interface (API) to this functionality is the Scheduler interface. It provides simple operations such as scheduling/unscheduling jobs and starting/stopping/pausing the scheduler.

JLayer Project

The JLayer is a real time MP3 decoder library which enables the intelligent medicine cabinet to play the sound of a siren when unauthorized people try to use it. The sound is played on a separate thread, letting the main code free to take care of the user's interactions.

MySQL Connector/J

MySQL provides connectivity for client applications developed in the Java programming language through a JDBC driver, which is called MySQL Connector/J. This library provides operations for Intelligent Medicine Cabinet insert update and delete information from its database. By using such library, one may also use the database via a web interface.

Medicine Cabinet Java Source Code

On the next sections all the Java classes and packages of the intelligent medicine cabinet will have their most important features described. Figure 43 below shows the system packages and the most relevant relationships among them.

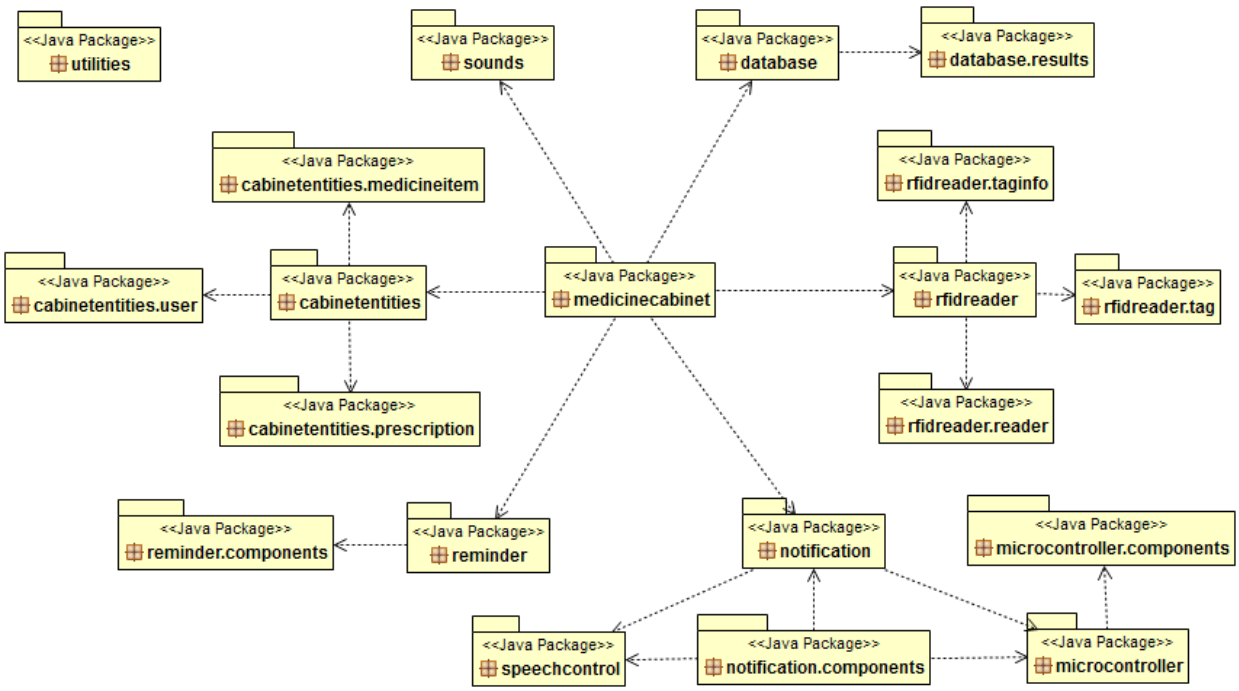

Figure 43: Medicine Cabinet Java Packages

Package <cabinetentities>

This package contains the three most important entities of the medicine cabinet. The class <MedicineItem> is the representation of the medicines in the cabinet. It contains methods and attributes to manage mostly information about the medicine package such as expiry date, quantity of pills and if the medicine is taken out or placed into the cabinet. Furthermore, it is enhanced with more general information provided from the package <cabinetentities.medicineitem> which is described below.

The class <Prescription> is the representation of the user's prescription. It manages if the prescription is taken out or placed into the cabinet and if it's still valid. The most important features of this class are: it manages user's medication, that is, if the medication is delayed, how many pills to take, the time on which the schedules should be taken, and the generation of reminders; it rebuilds the user's medication after the medicine cabinet is turned off and turned on again; and it calculates the next schedules of the medications. Furthermore, each prescription may have more than one medicine to have their medication managed, but a user may use only one prescription at a time. To enable each medication, the class on the package <cabinetentities.prescription> is necessary.

The class <User> represents the user and provides all the methods through which the user may interact with the cabinet, for example, to take a medicine or to check incompatibilities with medicines. An important characteristic of this class is that the information of medicines taken by the user is saved and is deleted (if necessary) when a medicine effect is over. To enable such "history" information, the class on the package <cabinetentities.user> is necessary.

Package <cabinetentities.medicineitem>

This package contains the classes which are necessary for the <MedicineItem> class. Its class <GenMedicine> is the representation of the general information of a medicine, that is, the information that is common to all the medicines, such as active ingredients, active ingredient group, incompatibilities with other medicines and dosage recommendations.

Whenever a medicine package is taken out of the cabinet, it is checked if the medicine should be taken or not and the reasons for that. The class <UserSession> manages this information by linking which user took the medicine out to the medicine package.

Package <cabinetentities.prescription>

This package contains only one class, which is necessary for the proper management of prescriptions. The class <MedOnPrescription> manages the information of a medicine which is necessary for its periodical medication, such as: duration of treatment, frequency of schedules, next schedule, and quantity of pills and dose to be taken. When a user has to take a medicine, the quantity of pills to be taken depends on the dose, for example: if the quantity to take is 2 and the dose is 2, the user has to take 4 pills. This class also manages the remaining doses to be taken and, as soon as the medication is over, it sets the medication state to 'finished', so that the <Prescription> class detects such event.

Package <cabinetentities.user>

This package also has only one class, the <TakenMedicine>, which manages the information important for a medicine that was taken by the user, such as, the quantity of pills taken, the time on which the medicine was taken and if a prescription was used for that purpose. With such information, it is possible to know when the effect of a medicine is over, for example.

Package <database>

This package contains all the classes necessary to make SQL queries to the medicine cabinet database. Its class <DataBaseHandler> provides the interface between the medicine cabinet and its database. It has all the methods to select, update, insert and delete data from the database. It also implements the class Thread and on its "run()" method, the medicine cabinet checks periodically if any medicine was announced as newly dangerous, to send to the rest of the system such information using the class <HarmfulInfo>, that wraps the information about the newly announced dangerous medicines.

Package <database.results>

To select and send information to the rest of the system in an easy way, the classes from the package <database.results> are necessary. Its classes take the raw results from the database and provides getters and setters for the information specified on Table 9.

Table 9: Classes for the database results

Class	Information
<HarmfulResult>	newly dangerous announced medicines
<MedGenResult>	general information of medicines
<MedInvResul>	medicine package attributes
<MedTakenResult>	medicines which were taken by the user
<MedToTakeResult>	prescription medicines attributes (e.g. time to take)
<PrescResult>	user's prescriptions
<UserResult>	user's birth date, language, and medicine allergies

Package <microcontroller>

This package contains the classes necessary to control the microcontroller functions. Its class <MicrocontrollerHandler> extends the Thread interface and, on its "run()" method, receives commands from the cabinet to blink and turn on/off the LEDs and check if the microcontroller detected the door open, sending the respectful information to the medicine cabinet. It also provides a method through which the medicine cabinet can check if the microcontroller is running properly. The other class, <MicroMessage>, provides the methods to turn on the LEDs or blink them with a pre-defined rate for a period of time. It is possible to send signals to each LED individually or use a combination of them.

Package <microcontroller.components>

This package contains the components which make the microcontroller actually work, that is, the microcontroller itself and the serial communication. Its class <Microcontroller> provides the interface to send commands to the microcontroller and to get data from it, for example, detect if the door is open and to check if it is working properly. A second class, <MicroSerialDriver>, realizes the serial communication between the PC and the microcontroller.

Package <notification>

This package contains the classes necessary to generate announcements to the user, either via speech output or visual signals. The <Notification> class builds a notification to be outputted after it is sent to the class <NotificationHandler>. A notification consists of LEDs

commands to the microcontroller; language ID of the language on which the sentences should be spoken; and the sentence (phrase) to be spoken.

In addition, the class <NotificationHandler> provides the methods to change the language spoken on the system and also implements the class Thread. On its "run()" method, messages from the microcontroller are received and notifications from the medicine cabinet are outputted. If, the language ID of the notification is different from the current language of the system, the spoken language is also changed on such method.

Package <notification.components>

This package contains the pre-defined notifications, either with visual signals as well as with sentences to be spoken. With the class <Notifications> the system uses pre-defined structured phrases to dynamically generate valid speech output to the user, that is, speech output with the user name or with dynamic information from each interaction with the medicine cabinet. In addition, the class <VisualSignals> has methods with the same names as the methods of the class <Notifications>, but it uses pre-defined LED combinations to generate visual signals to the <Notifications>.

Package <reminder>

This package contains the classes to enable the medicine cabinet with reminders. It can be used to remind the user of the medication schedules or the system about the time when the effect of a medicine is over. The class <Reminder> represents a reminder unit with the following attributes/methods: ID of the message formed from its hashcode; when the message should be delivered to the system (when to remind the user); a schedule which contains information for the <ReminderHandler> on how to trigger the reminder; a stop and initiate schedule methods to cancel the triggering of the reminder or sent it to the Quartz thread, which waits the reminder time to be achieved to trigger it.

Furthermore, the class <ReminderHandler> has a list of all the active reminders and manages them. With this class it is also possible to check if the reminder is no longer necessary (e.g., if the user took the medicine) and remove it from the list of active reminders.

Package <reminder.components>

This package contains what basically forms a reminder. The class <ReminderJob> is the core unit which is triggered by the Quartz Scheduler library. When the time for the reminder

is reached, it is the <ReminderJob> which is run on the system. This job basically prints out on the console the message to be delivered and sends this message via a <QueueUnit> channel to the system.

The class <ReminderMessage>, in turn, wraps the message of the reminder and provides getters to all its fields, such as when to be triggered, target user, which medicine the information is about, if it is under prescription, how many pills to take and etc.

Package <rfidreader>

This package has all the classes necessary to use the RFID reader. Its class <RFIDReader> provides the methods to: detect the IDs of the tags within the reader range; get data from the detected IDs; and write data to the tags. These methods need the <RFIDSerialDriver> to function properly and they are used on the <RFIDReaderHandler>.

The class <RFIDReaderHandler> implements the Thread interface. On its "run()" method, the tags IDs are requested periodically. If there is a new tag in the reader range, its data is retrieved and sent to the rest of the system. The same thing happens when it is detected that a tag is not present anymore, in this case, only the tag ID is sent to the system.

Package <rfidreader.reader>

This package contains the elements to establish the communication with the reader and the RFID commands to be used. The first class, <RFIDReaderMessages>, has the commands to read the inventory (detect the IDs in range), to get data from a tag (request tag data) and to write data to the tag. And the second one, <RFIDSerialDriver>, as the name says, is in charge of establishing the communication with the reader via a serial interface. It receives serial data in an interrupt-driven fashion and, in case the windows buffer of the serial communication is messed up because the operating system is not for real time applications, it generates an error message defined specifically for the medicine cabinet.

Package <rfidreader.tag>

This package has the classes to handle byte information of the tags in hexadecimal format and to keep track of the status of the tag. The class <Tag> has the methods to calculate the CRC16 of the messages sent to the reader as well as methods to convert tag data from hexadecimal to String and vice versa.

If the tag is in an unstable region or bad positioned, so that the reader cannot retrieve its information properly, the class <TagStatus> stores information about this problematic situation and, through it, the medicine cabinet can tell the user that such tag should be repositioned.

Package <rfidreader.taginfo>

This package contains the classes that wraps the information from the RFID tags, encapsulates them into objects, and provides getters for their attributes. Its classes contains the following information: class <MedTagInfo> is used to get information of the medicines tags; class <MedToTake>, to get information of the medicines to be taken, which are stored on the prescriptions tags; the class <PrescTagInfo>, to get information of the prescription tags; class <GenTagInfo> is used for the general information of the tags, for example, if they were added, removed or responded with error to the reader; and the class <UserTagInfo>, which provides getters to get information of the users' tags.

Package <sounds>

This package has the class to play MP3 sounds, <CabinetPlayer>. It has the methods to create an object which plays a siren sound in two situations, when no user is logged in and a medicine is taken out, and when a child is logged in, but no adult is logged in along with them. This class is important to warn that only authorized people should use the medicine cabinet.

Package <utilities>

This package contains some classes that are used by many other classes on the system. Its class <Clock> is in charge of making operations related to date, such as to calculate the difference in seconds, hours and days between two dates, and provide string format of dates to the <SpeechControl> class.

The class <QueueUnit> is used by other classes to queue objects. It, used alone, or being used by the <ThreadInterface> class, provides the most important way through which all the threads of the medicine cabinet communicate to each other, in other words, sending messages in a queued style to each other. The <ThreadInterface> is used by all the threads on the system to send messages to each other. It provides an "Rx-Tx" channel, so that threads

may send messages to each other whenever it is necessary, without the need of waiting the channel to be free.

In addition, there is the class <StackUnit> that has the same structure of <QueueUnit>, but it stores the messages (objects) in a stack order. It is mainly used to manage which users should be active at a time.

Package <speechcontrol>

This package contains the classes for the speech output of the medicine cabinet. The first one, <LanguagesIDs>, has the ID languages to be used by the medicine cabinet. Each ID is also an index to the <Notifications> class messages. The second one, <SpeechControl>, provides all the functionalities to allocate a speech synthesizer to the medicine cabinet using the installed voices on the system. It is possible to change the language spoken as well as set some attributes, such as speaking rate and volume.

Package <medicinecabinet>

This package contains the most important classes of the medicine cabinet: the classes which make all the logic and sequencing of the system. The system was programmed in such a way that most of the sequencing of processing should be made on this two classes, which increases the number of code lines, but makes it easier to understand the whole logic of the medicine cabinet.

Class <MedicineCabinet>

It is the most important class for the intelligent medicine cabinet on the point of view of the system logic, as it works as a front-end to the external inputs to the system and coordinate all the work of the other threads.

It has the following characteristics:

- Initialization of all the threads.
- Initialization of the first system procedural logics, for example, checking which cards, medicines or prescriptions that were previously in the cabinet before it was turned off.
- It checks if the door is open and makes an action upon this event.
- It checks if there is any reminder triggered and, if so, the user is informed about it.

- It checks if there is any message from the RFID reader, that is, if a tag was added ore removed from the read range.

One tag may have three statuses and the following actions are performed:

- Tag added – it is checked if the tag was previously on the system memory or if it is completely new. According to the type of tag, an action is made, for example, logging in a user or setting a medicine as "in the inventory".
- Tag removed – if the tag represents a user, they are logged out. If it refers to a medicine, its ID is sent to the list of taken out medicines and a session for the current active user's actions is created upon this taken out event. If the tag represents a prescription, it is also added to a taken out list of prescriptions. These lists are managed by the <Inventory> class, which is discussed later.
- Tag with problem – if a tag is on an unstable region or position and it is periodically removed from and added to the system, it means that it might be under problematic reads. The user is then informed about this situation, to reposition the tag.

Class <Inventory>

It is the most important class on the point of view of entities interactions: user, medicine and prescription. It has a list of users, prescriptions and medicines in the medicine cabinet inventory and, by combining their information, it performs the following actions, when a medicine or prescription is taken out:

- Checks if the current user is authorized to use the cabinet or not.
- Checks if user has incompatibilities against the medicine (allergy, for example).
- Checks if the taken out medicine has a prescription for it, so that the medication should be managed, for example, what to do in case of a delayed schedule, the calculation of schedules and the right dosage to be taken.
- Checks if a medicine is harmful, that is, if it is expired or was announced as newly dangerous.

When the medicine is away from the cabinet for a certain period of time or it is placed into the cabinet again, the information of the interactions mentioned above is already saved on the user's session upon the medicine. Then the following actions are performed:

- Checks if the medicine should be taken or not. If it should be taken, it is considered that the user took it if the medicine was away for a certain period of time. If the medicine has prescription, the next schedule is calculated and the reminder for it is created.
- Checks if the medicine or prescription were away for a period of time, on which they are considered as removed.

All the interactions mentioned above are made on the <Inventory> "run()" method, which implements the java Thread interface. Besides of having a thread running on the system, <Inventory> also provides the methods used by the class <MedicineCabinet> to add and remove user's, prescription and medicine information on the system.

Appendix D- Software Component: Database

Intelligent Medicine Cabinet Database

The Intelligent medicine cabinet has a database called *"medicine_cabinet"* with the following tables:

- *"harmful_medicines"* – this table contains information about the medicines which are announced as newly dangerous.
- *"medicines"* – this table has the general information about the medicines, which are common to all of them.
- *"medicines_inv"* – this table stores the specific information about the medicines which are currently being used in the cabinet.
- *"medicines_taken"* – the information related to the medicines taken by the user are stored on this table.
- *"meds_to_take"* – this table has information on how the medication for each medicine should be done, it has a foreigner key to the table *"prescriptions"*
- *"prescriptions"* – this table has the general information about the prescription, such as the name of the user and list of medicines.
- *"prescs_register"* – when a medicine has its medication finished, the related information is saved on this table.
- *"users"* – this table stores information about the users.

Database table <harmful_medicines>

Table 10: Database table of *"harmful_medicine"*

#	Name	Type	Collation	Attributes	Null	Default
1	**medicine_name**	varchar(50)	utf8_general_ci		Yes	*NULL*
2	**batch_number**	varchar(8)	utf8_general_ci		Yes	*NULL*
3	**checked**	tinyint(1)			No	0

Table 10 shows the structure of the table. When a medicine is announced as newly dangerous, it is saved with the field "medicine_name". If only a batch of the medicine is dangerous, it is saved on the field "batch_number". The field "checked" is used by the medicine cabinet Java program, which periodically reads this table to verify if there is any unchecked row, if so, such information should be updated on the system, so that, on the next time a user takes out a medicine, they should be informed about that.

Database table <medicines>

Table 11: Database table of *"medicines"*

#	Name	Type	Collation	Attributes	Null	Default
1	**medicine_name**	varchar(50)	utf8_general_ci		No	*None*
2	**needs_prescription**	tinyint(1)			No	*None*
3	**dose**	varchar(50)	utf8_general_ci		Yes	*NULL*
4	**active_ingredients**	varchar(100)	utf8_general_ci		No	*None*
5	**active_ingredient_group**	varchar(50)	utf8_general_ci		No	*None*
6	**incompatibilities**	varchar(100)	utf8_general_ci		Yes	*NULL*

Table 11 shows the structure of this table. The fields have self-explanatory names, but two of them should have their meaning explained:

- dose – it represents the maximum daily quantity of pills that should be taken by each user group (child, adult and elderly). This field is in the form "C:A:E", where "A" represents the dose for children, "A" for adults and "E" for elderly. If the value for one of these groups is set as zero, the doctor should be consulted before taking this medicine. For example, "0:4:3" means that children shouldn't take this medicine without checking the right dosage with a doctor.

- incompatibilities – this field has the name of medicines which are incompatible to the current medicine row.

Database table <medicines_inv>

Table 12: Database table of *"medicines_inv"*

#	Name	Type	Collation	Attributes	Null	Default
1	**id**	varchar(16)	utf8_general_ci		No	*None*
2	**medicine_name**	varchar(50)	utf8_general_ci		No	*None*
3	**expiry_date**	date			No	*None*
4	**batch_number**	varchar(8)	utf8_general_ci		No	*None*
5	**quantity**	int(3)			No	*None*
6	**in_inventory**	tinyint(1)			No	*None*

Table 12 shows the structure of the table. Some names should have their meaning explained:

- id – this is the ID taken from the RFID tag.
- quantity – this is the quantity of pills remaining in the medicine package.
- in_inventory – this boolean field tells if the medicine package is currently in or out of the medicine cabinet.

Database table <medicines_taken>

Table 13: Database table of "medicines_taken"

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**register**	int(4)			No	*None*	AUTO_INCREMENT
2	**user_name**	varchar(50)	utf8_general_ci		No	*None*	
3	**medicine_name**	varchar(50)	utf8_general_ci		No	*None*	
4	**time_taken**	datetime			No	*None*	
5	**time_to_take**	datetime			Yes	*NULL*	
6	**dose_taken**	int(2)			No	*None*	
7	**by_prescription**	varchar(16)	utf8_general_ci		Yes	*NULL*	

Table 13 shows the structure of the *"medicines_taken"* table. It stores information about the medicines which are taken by the users. Besides most of them are self-explanatory fields, some of them need to be explained:

- register – this is an auto-increment field and is automatically update by the database when a new row is inserted.
- by_prescription and time_to_take – if the medication was taken because of the use of a prescription, the ID of the prescription is stored in the "by_prescription" field and the time that the medicine should be taken in "time_to_take". That way is possible to know if the user is taking the medicine on the right time by consulting the database.

Database table <meds_to_take>

Table 14: Database table of *"meds_to_take"*

#	Name	Type	Collation	Attributes	Null	Default
1	**prescription_id**	varchar(16)	utf8_general_ci		No	*None*
2	**medicine_name**	varchar(50)	utf8_general_ci		No	*None*
3	**duration**	int(3)			No	*None*
4	**frequency**	int(3)			No	*None*
5	**remaining_doses**	int(3)			No	*None*
6	**next_taking**	datetime			Yes	*NULL*
7	**next_dose**	int(3)			No	*None*
8	**quantity**	int(3)			No	*None*
9	**delay_action**	int(3)			No	*None*
10	**acceptable_delay**	int(3)			No	*None*
11	**start**	datetime			Yes	*NULL*
12	**end**	datetime			Yes	*NULL*

Table 14 shows the structure of the most complex table for the system: medicines under medication. It has the following fields:

- prescription_id – this is the ID of the prescription to which the medicine is linked (foreigner key).
- medicine_name – the name of the medicine

- duration – it represents the total number of days that the medication lasts.
- frequency – if the doctor says that the user should take the medicine every 6 hours, the value “6” should be stored on this field.
- next_taking – it is the time for the next medication schedule
- next_dose and quantity – the previous tables showed “dose” as meaning the quantity of pills taken or to be taken, but in the “meds_to_take” table, it has a slightly different meaning. Instead of the quantity of pills, it means a multiplier for the field “quantity”. For example, if the field “next_dose” has the value “2”, which means double dose and the field “quantity” has the value “2”, then, the actual quantity of pills to be taken by the user on the next schedule is 4 pills (i.e., 2x2).
- delay_action – this is the code about what should be done on the next schedule if the user misses or is delayed to the current one. This code has values from 0 to 3: “0” means that double dose is not allowed; “1”, that double dose is allowed; “2”, that a new cycle should be started; “3”, that the medicine should be taken as soon as possible.
- acceptable_delay – this field represents the percentage of the frequency on which there is no problem in taking the medicine. After this limit is achieved, the delay_action field should be used to compute the next schedule. As an example, if this field is set as “50”, it means that after 50% of the frequency, a “delay_action” should be performed to calculate the next schedule. If it should be taken every 4 hours, the user would have two extras hours to take it without any problem.
- start – this field stores the date when the medication starts.
- end – this filed stores the date when the medication is finished.

Database table <prescriptions>

Table 15: Database table of *"prescriptions"*

#	Name	Type	Collation	Attributes	Null	Default
1	**id**	varchar(16)	utf8_general_ci		No	*None*
2	**repeat**	tinyint(1)			No	*None*
3	**user_name**	varchar(50)	utf8_general_ci		No	*None*
4	**medicines**	varchar(50)	utf8_general_ci		No	*None*
5	**active**	tinyint(1)			No	*None*
6	**date_activated**	datetime			Yes	*NULL*

Table 15 shows the structure of the table. It has the following fields:

- id – the ID taken from the prescription RFID tag.
- repeat – that means, if the prescription has medicines which could be taken for a period of time, have the medication paused and the restarted again. If a prescription has medicines that, once the medications are over it shouldn't be used again, this field should be flagged as false.
- user_name – the user's name for this prescription.
- medicines – it stores the names of the medicines for the prescription.
- active – if all the medicines medications for the prescription are finished or have not started yet, this field is set as "false", otherwise, if the medication are not finished, this field is set as "true".
- date_activated – this field stores the data of the first medicine of the prescription that had its medication started. Combining "active" and "date_activated" it is possible to know if the medication for the prescription has not started, has been started or is already finished.

Database table <prescs_registers>

Table 16: Database table of *"prescs_registers"*

#	Name	Type	Collation	Attributes	Null	Default
1	**prescription_id**	varchar(16)	utf8_general_ci		No	*None*
2	**medicine_name**	varchar(50)	utf8_general_ci		No	*None*
3	**start**	datetime			No	*None*
4	**normal_end**	datetime			No	*None*
5	**end**	datetime			No	*None*

Table 16 shows the structure of *"prescs_registers"* table. For each medicine that has its medication finished, a register is saved on this table. The field "normal_end" and "end", when compared, tell if the user took the medication correctly.

Database table <users>

Table 17: Database table of *"users"*

#	Name	Type	Collation	Attributes	Null	Default
1	**id**	varchar(16)	utf8_general_ci		No	*None*
2	**user_name**	varchar(50)	utf8_general_ci		No	*None*
3	**date_of_birth**	date			No	*None*
4	**language**	varchar(50)	utf8_general_ci		No	*None*
5	**allergies**	varchar(50)	utf8_general_ci		Yes	*NULL*
6	**incompatibilities**	varchar(50)	utf8_general_ci		Yes	*NULL*

Table 17 shows the table which stores the information about the user. It is important so that the medicine cabinet detects if a user had already used the cabinet or not.

Appendix E- Software Component: Web Service

Web Service

The web service is in charge of making the information of the medicine cabinet available to the web. As it can be seen on Figure 44 (WSDL document), it has four main groups of services to retrieve information of the following database tables: recalled medicines, medicine packages present on the cabinet, user medications, user prescriptions and user personal information.

The services may be requested using the HTTP operation GET that returns information of the tables and allows writing to them via POST. As example, to request general information of the medicines, a typical operation would be (on local host): "http://localhost/MCWebservice/index.php?r=gen_info&has_inc=true&pp". The web service allows the parameters to be written in any order ("pp" could have be written in the beginning of the request). Although each table has its specific operations, some parameters are present on all of them, such as: "r", which is the type of information requested (which table); and "pp", which indents the output (making it easier to read).

Table 18 shows the parameters of each operation (excluding the "pp" parameter).

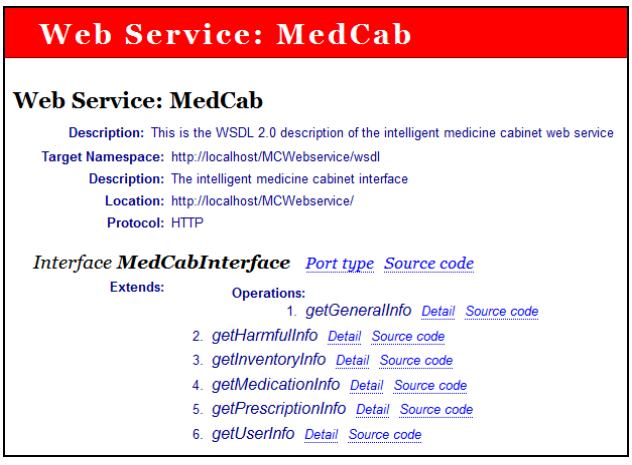

Figure 44: Web service general description

Table 18: Web services

Type	Parameters	Values description
Read general information of medicines (r = "gen_info")	has_inc	if medicine has any incompatibility
	Ing	medicine active ingredient
	Inc	an incompatible medicine with the searched one
	Med	medicine name
	Presc	if medicine needs a prescription to be taken
	ing_grp	medicine active ingredient group
Read harmful medicines information (r = "harmful_medicine)	Med	medicine name
	Batch	medicine batch number
Read information inventory medicines (r = inventory)	exp_on	specific expiry date
	exp_before	medicine expires before specified date
	exp_after	medicine expires after specified date
	Med	"all" medicines or a specified one
	Id	"all" ids or a specified one
	in_inv	only medicines that are on the cabinet
Read medications information (r = med_info)	Med	medication with the specified medicine name
	User	medication for the specified user name
	Act	if the medication is active (i.e. has started)
	All	returns all the medications
	Next	medications scheduled for a specified date
Read prescriptions information (r = prescription)	act_on	date on which the prescription was activated
	act_after	date after prescription activation
	act_before	data before prescription activation
	Act	any prescription that is active
	Repeat	if prescriptions may be reused or not
	User	user's name
	Id	"all" prescription IDs or a specified one
	Lang	prescription language
Users' information (r = user)	Birth	specific user's birthdate
	younger_than	user is younger than the specified age
	older_than	user is older than the specified age
	Age	user has an exact age
	has_inc	if user has any medicine incompatibility
	has_alle	if user has any allergy
	Incomp	user has the specified incompatibility
	Alle	user has the specified allergy
	User	user's name
	Id	"all" users' IDs or a specified one
	Lang	the user's language

As examples of outputs, Figure 45 shows the answers formatted with the “pp” parameter (pretty print format) of requests to get general information about medicines.

Output: type *genInfo* *Source code* type *string*

That is a success JSON output example:


```
{
"success": 1,
"medicines": [
{
"medicine_name": "Citalopram",
"needs_prescription": "0",
"dose": "2:4:3",
"active_ingredients": "Citalopram",
"active_ingredient_group": "SSRI",
"incompatibilities": "Ibuflam"
},
{
"medicine_name": "Ibuflam",
"needs_prescription": "0",
"dose": "2:4:3",
"active_ingredients": "Ibuprofen",
"active_ingredient_group": "Nichtsteroidales Antirheumatikum",
"incompatibilities": "Citalopram"
}
]
}
```


JSON output when valid search returns no result:


```
{
"success": 0,
"message": "No medicine found"
}
```


JSON output when no valid parameter is used:


```
{
"success": 0,
"message": "At least one refinement must be present"
}
```


JSON output for an invalid parameter value:


```
{
"success": 0,
"message": "Invalid parameter value"
}
```


Figure 45: Web service response for medicine general information

Appendix F- Reading and Writing Tag Information

As the prototype was designed solely to interface with the user assuming that all the tags have their information previously saved to them, third-party software is necessary to write and edit information on the tags. A GUI software was written in Java with this purpose. With this GUI the doctor or caregiver may write information to the users' cards, medicine packages and prescriptions.

It is important to highlight that this software can only edit information of one tag at a time. Figure 46 shows the tab for the edition of medicine tags. By using the "read tag info" button, the information of the medicine package is read and written to the text fields. By using the button "=>", the fields values from the reading box are copied to the writing fields and may be edited and saved to the tag by using the button "write info to tag". From the status box on the bottom, it is possible to know if the tags were read/written successfully or if there was any error. The same logic applies for Figure 47 and Figure 48, but at the latter it is possible to edit two medicines of the prescription list at the same time (this is the maximum number of medicines allowed for each prescription).

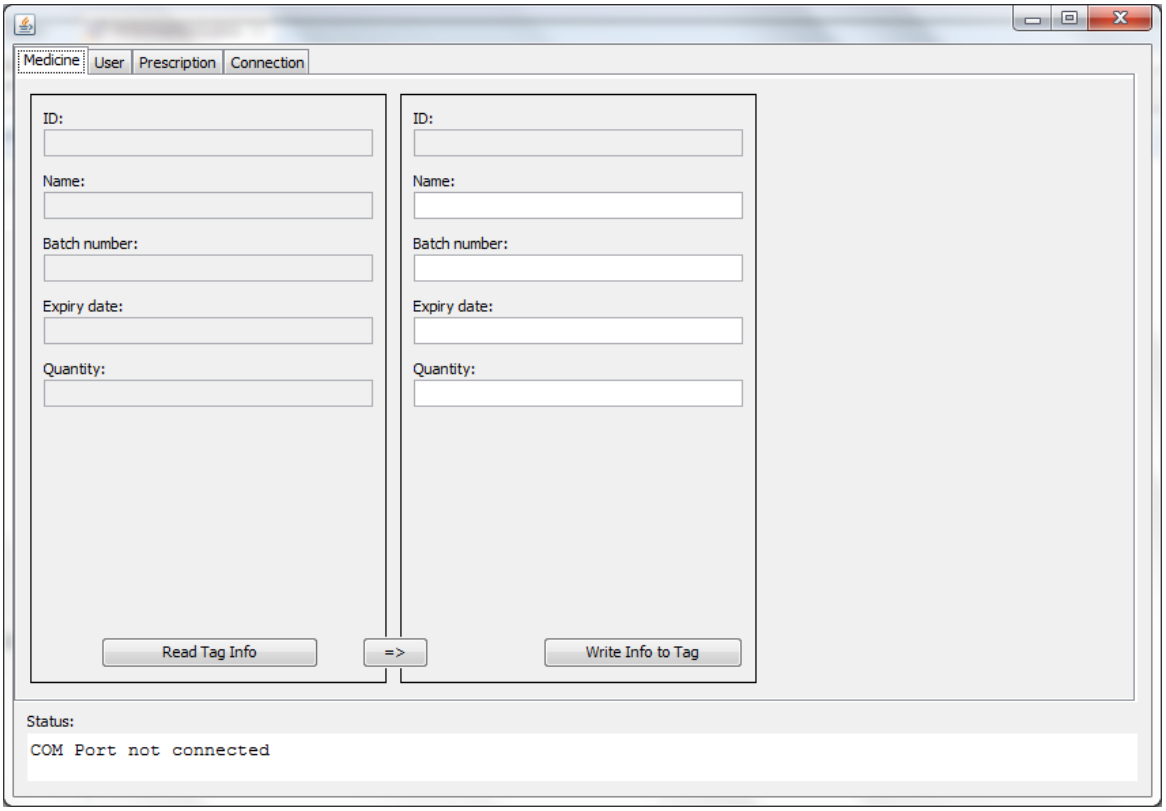

Figure 46: GUI screen to edit medicine tags

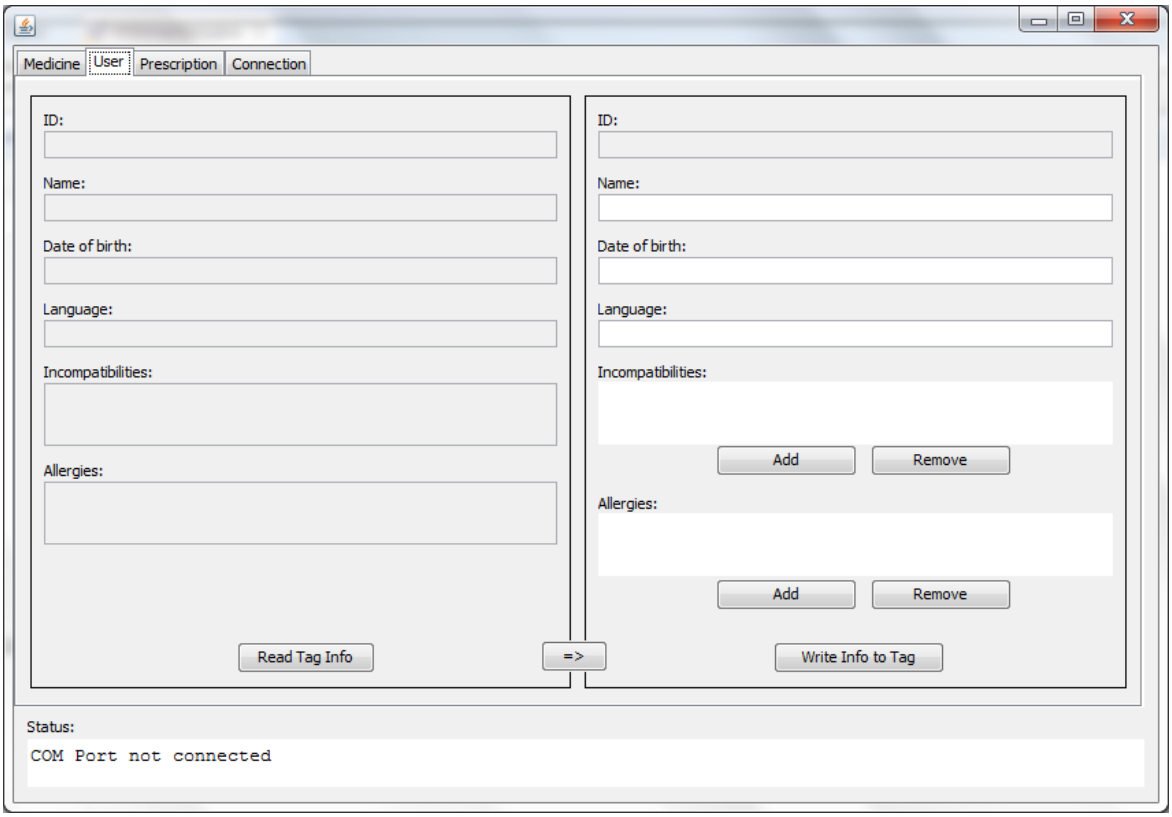

Figure 47: GUI screen to edit user cards

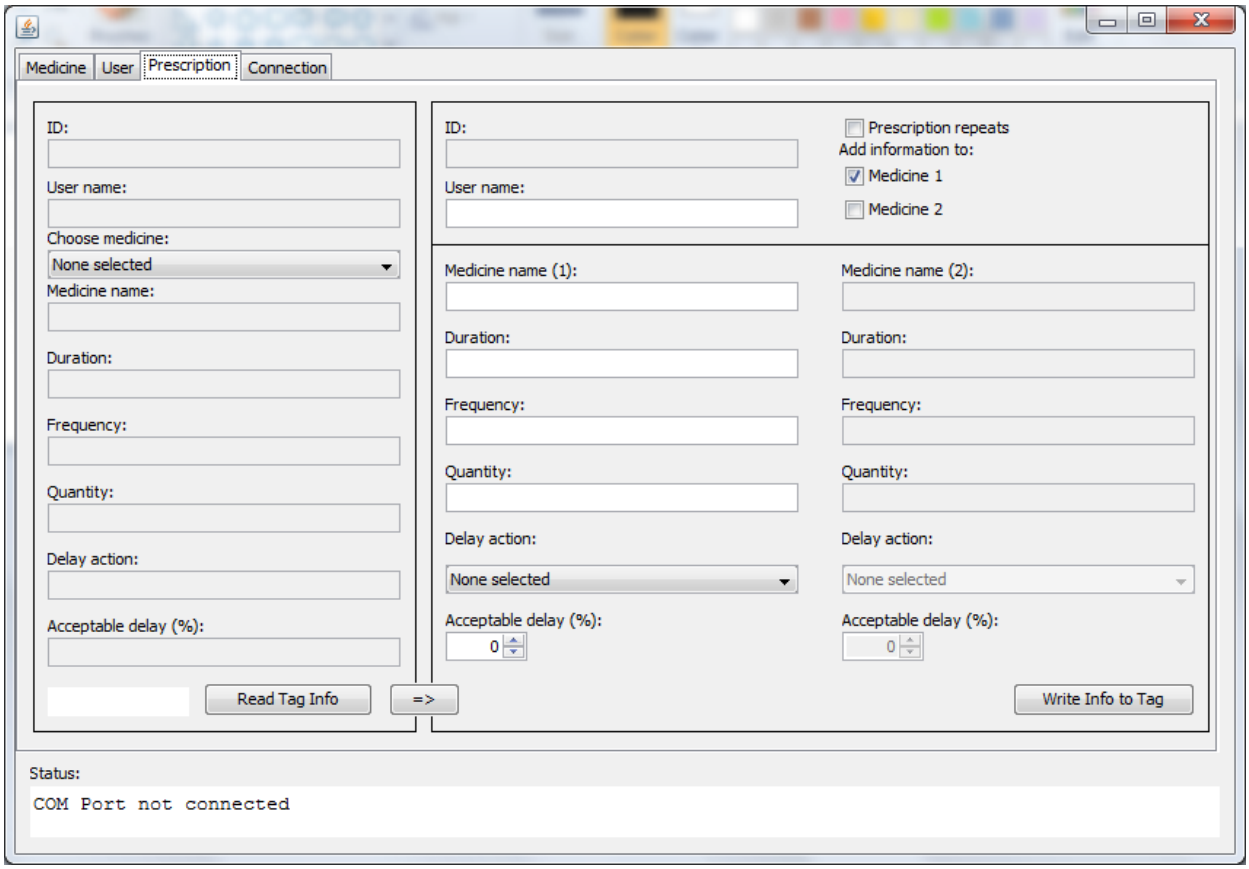

Figure 48: GUI screen to edit prescriptions

To edit the tags information, it is necessary first to set a serial communication with the RFID reader (Figure 49). It is necessary that the user type in which COM port is in use by the reader. On this screen there is another very important functionality to keep the quality of readings: the "calibrate" button. When it is used, the system will test which delay value between each reading request is less prone to error while identifying the tags IDs. From 5 to 100ms, with steps of 5ms, every delay value is tested 10 times and their readings accuracy and total average delays are stored. In the end, the value with lowest total average delay and most accurate is chosen to posterior usage. This value can also be used on the prototype, as it is an optimal delay between RFID readings. That way, it is possible to have a calibrated value independent of the computer on which the prototype or GUI software runs.

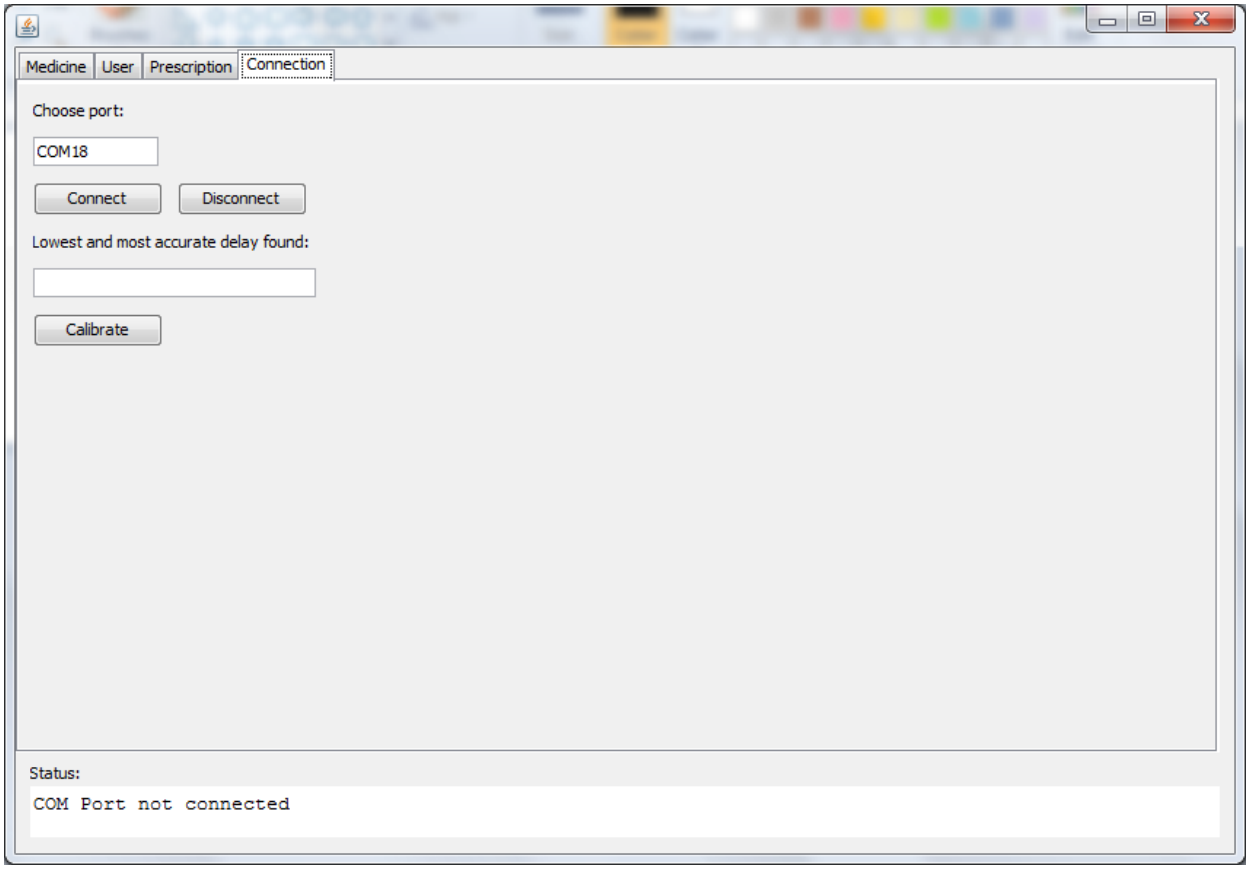

Figure 49: GUI screen to set serial communication

Appendix G- Prototype Troubleshooting

This section covers the possible events that are related to problems with the medicine cabinet. During the development of the medicine cabinet, three classes of events are known to happen: those that are mistakenly considered as error; those that induce the system into an error situation, but it is not a malfunction of the system; and those that are, in fact, an error situation.

The following topics discuss about the events that are mistakenly considered as an error:

1. Sometimes, when an item (user card, prescription or medicine) is taken out system announces something to the user. This is not an error and it happens because, depending on the interaction, the system has to query information from the database twice or more times.
2. If many users are logged in/out of the system, at the same time or one after the other, the system will give speech output for each one of these actions. As an example, if there are only three users using the cabinet, let's say user "A", "B" and "C". If they are logged one after the other in such order, the medicine cabinet will say that "A" has logged out or placed into the cabinet, there is a delay before the
3. and that "B" is active, then that "B" has logged out and that "C" is active and, finally, that "C" has logged out and that there is no user logged in anymore. For a moment, the system announced that "B" and "C" were active, even though their cards were taken out. It does not mean that the system computed something wrong, it indeed computed the sequence right. That happens because the logic of the medicine cabinet is way faster than the speech output. That way, when the first announcements were still being spoken, the system had already computed all the other events, but the sequence of messages to be spoken had already been sent to the notifications queue. The system does not wait for a second use to log out to compute something, but it responds to each event separately.
4. If two or more items (user card, prescription or medicine) are taken out or placed into the cabinet, the system will treat each event uniquely. That means that if the user, for example, takes out two medicines and the first one is under medication

with a prescription, the announcement about the “removal” of the second will only be spoken after all the message for the first one are spoken, which contains extra medication information because of the prescription.

Situations that may induce the system to error:

1. If one item was not taken out of the cabinet, but the system says it was taken out and later placed into the cabinet, even for consecutive times, that means that the tag is not in a good position to be detected and, for some readings, the system does not recognized it within its range. This may case many announcements in a row (saying “taken out” and “placed into”) until the system recognizes this behavior and warns the user. The problem is that the system only classifies this as a problem if such behavior repeats at least 3 times within 30 seconds. If it happens every 30 seconds, for example, it won’t be announced as a problem and the user has to realize that himself and reposition the tag.
2. This “detection” problem cited above may become generalized when many tags are on the same problematic situation. The announcements of each one (“taken out” and “placed into”) will be spoken until the system recognizes such behavior. The problem, again, is if this behavior does not happen for 3 times within 30 seconds, than it will not be classified as an error.
3. It is possible that one tag is not having problems to be detected, but, as soon as another tag is put in a problematic position, the other tag begins to have problems to be detected too. If the behavior repeats within those 30 seconds, one of them (or both) will be announced as problematic and the user will be instructed on repositioning it. If the system does not recognize it as a problematic situation, the user has to realize it himself and reposition the tag.

Situations that are error:

1. If the system still announces when an item is placed into the cabinet or taken out of it, but has stopped announcing information about their interactions, for example, if the user could take a medicine, then the thread in charge of checking their interactions had problems and the system has to be restarted. Using the card to

restart only the application should suffice, otherwise, if the problem persists, the card to restart the operating system must be used.

2. If the system stops speaking at all, it is possible that the main thread is dead for some reason, the RFID reader is not properly connected or that the synthesizer engine is not working properly. It should be checked if the reader is properly connected to the PC or to its antenna. If it is properly connected, using the card to restart only the application should suffice. If the problem persists it may be a problem with the synthesizer engine and the card to restart the operating system has to be used.
3. If the system stops generating visual signals, either the microcontroller thread is dead for some reason or there is a problem with its connection to the PC. If the microcontroller thread is dead, the rest of the system will still work properly. To fix it, it should be checked if the USB cable of the microcontroller is properly connected. Using the card to restart the application should suffice.

Appendix H- List of Publications

- Claudio E. M. Gomes; Farzan Yazdi; Vicente F. Lucena Jr.; Peter Göhner. “Um Armário de Medicamento Inteligente Desenvolvido para Aumentar a Eficiência da Medicação”. SBAI – Simpósio Brasileiro de Automação Inteligente, Fortaleza, 2013.

- Claudio E. M. Gomes; Farzan Yazdi; Vicente F. Lucena Jr.; Peter Göhner. “An Intelligent Medicine Cabinet Proposed to Increase Medication Adherence”. IEEE Healthcom, Lisbon, 2013.