INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE - IFSUL, CÂMPUS PASSO FUNDO
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

VANESSA LAGO MACHADO

GEOLOCALIZAÇÃO *INDOOR*: UM ESTUDO DE CASO USANDO RFID NA PLATAFORMA ARDUINO

Orientador (a): Prof. Esp. José Antônio Oliveira de Figueiredo

PASSO FUNDO, 2013

VANESSA LAGO MACHADO

GEOLOCALIZAÇÃO *INDOOR*: UM ESTUDO DE CASO USANDO RFID NA PLATAFORMA ARDUINO

Monografia apresentada ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-Rio-Grandense, Câmpus Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador (a): Prof. Esp. José Antônio Oliveira de Figueiredo

PASSO FUNDO, 2013

VANESSA LAGO MACHADO

GEOLOCALIZAÇÃO *INDOOR*: UM ESTUDO DE CASO USANDO RFID NA PLATAFORMA ARDUINO

Trabalho de Conclusão de Curso aprovado em ____/____/____ como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet

Banca Examinadora:

_____

Prof. Esp. José Antônio Oliveira de Figueiredo
Orientador

_____

Prof. Msc. Anubis Graciela de Moraes Rossetto
Avaliador

_____

Prof. Msc. Lisandro Lemos Machado
Avaliador

_____

Prof. Dr. Alexandre Tagliari Lazzaretti
Coordenador do Curso

PASSO FUNDO, 2013

À Deus, por sempre iluminar meu caminho
e aos meus pais, pela compreensão e o estímulo
em todos os momentos.

AGRADECIMENTOS

Primeiramente gostaria de agradecer à Deus, por sempre iluminar meu caminho, mostrando que tudo tem seu tempo certo.

Agradeço aos meus pais, Mário e Sônia, pelo apoio e estímulo a mim fornecido. A minha mãe, Sônia, pela sua batalha constante pelo estudo dos filhos, lutando sempre pra que fossemos o melhor possível.

Agradeço as pessoas que me acolheram e me ajudaram em momentos difíceis da minha vida Nery e Elizabeth e meu namorado Marcos C. Teixeira.

Agradeço a todos os professores do curso superior de Tecnologia em Sistemas para Internet do IFSul, em especial ao meu orientador, José A. O. Figueiredo, por todo apoio e confiança a mim concebida. Agradeço também ao professor Juliano Menegaz, que infelizmente não se encontra mais entre nós, porém deixou junto a sua memória muitas lições de vida. Aos meus colegas que me acompanharam no percurso da faculdade, em especial Guilherme A. Borges, Julia C. Passamani e Nayne M. Dionisio, por todas as experiências juntas vividas.

Agradeço as minhas amigas Kimberli T. Loss e Marina H. Machado por todo apoio nessa jornada e por sempre estarem disposta a me ajudar.

Por fim, agradeço a todos os familiares, amigos e a todas as pessoas que de alguma forma colaboraram na minha construção de vida e em especial nesta caminhada em busca do conhecimento.

“Descobrir consiste em olhar para o que todo mundo está vendo
e pensar uma coisa diferente”.
Roger Von Oech

RESUMO

O presente trabalho apresenta um estudo referente à tecnologia de geolocalização, com enfoque em ambientes *indoors.* Assim, o objetivo é propor, através de um estudo de caso, uma alternativa para localização nestes ambientes, atendendo os requisitos de desenvolvimento de aplicação de baixo custo e disponível através de uma interface amigável, utilizando como tecnologia de identificação o RFId sobre a arquitetura de prototipagem arduino.

Palavras-chave: Geolocalização; RFID; Arduino.

ABSTRACT

This paper presents studies relating to geolocation technology, focusing on indoors environments. The objective of this study is to propose an alternative location for these environments, attending the low cost requirements of the application development, available through a user-friendly interface, using as identification the RFID technology on the Arduino architecture.

Keywords: Geolocation; RFID; Arduino.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ANATEL - Agência Nacional de Telecomunicações, p. 29

AP - *Acess Point*, p. 17

BPS - bits por segundo, p. 42

CRUD - *Create, Read, Update and Delete*, p. 58

DAO - *Data Access Object*, p. 46

EPC - *Electronic Product Code*, p. 25

ER - Entidade Relacionamento, p. 49

E/S - entrada e saída, p. 31

GPS - *Global Positioning Systems*, p. 14

HF - *High Frequency*, p. 23

ID - *Identify*, p. 20

IDE - *Integrated Developement Environment*, p. 32

IFSul - Instituto Federal Sul-rio-grandense, p. 13

IP - *Internet Protocol*, p. 14

ISO - *International for Standardization*, p. 25

LF - *Low Frequency*, p. 24

LPS - *Local Positioning System*, p. 15

MAC - *Media Access Control*, p. 14

RF - Radiofrequência, p. 15

RFId - *Radio Frequency Identification*, p. 13

SQL - *Structured Query Language*, p. 51

UHF - *Ultra High Frequency*, p. 23

USB - *Universal Serial Bus*, p. 30

SUMÁRIO

1 INTRODUÇÃO

O conceito de geolocalizar pessoas (assim como objetos) não é novo e apresenta uma enorme gama de técnicas e tecnologias para tal função. Estas técnicas já foram utilizadas em diversas situações, como, por exemplo, guerras como uma ferramenta contra o inimigo.

Porém, mesmo com toda a evolução destas tecnologias, existem ambientes em que sua eficiência ainda não é satisfatória. Ambientes *indoors* (fechados) necessitam de uma precisão extremamente grande e a margem de erro que pode ser aceita, em relação a real localização, deve ser significativamente pequena, pois baixa precisão pode gerar erros consideráveis em relação à posição exata do objeto.

Nesse projeto foi desenvolvida uma aplicação de geolocalização direcionada a ambientes *indoors*, buscando tratar a necessidade de maior precisão possibilitando, assim, a localização de objetos ou pessoas pré-determinadas em ambientes empresariais (assim como residenciais, entre outros ambientes fechados), usando tecnologias que possuam custo relativamente baixo.

1.1 MOTIVAÇÃO

Atualmente, a geolocalização em ambientes fechados ainda se mostra deficiente, pois há necessidade de estudos que tornem esta tecnologia mais precisa. Além disso, há necessidade de redução nos custos da implantação de tal tecnologia, possibilitados por novos recursos existentes em *hardware*. Outra demanda decorrente destes sistemas é a implantação de uma interface de fácil manuseio a usuários leigos, os quais farão uso deste.

Assim, torna-se um desafio projetar um sistema que através da necessidade de geolocalização *indoor*, satisfaça todos os pontos a serem melhorados, conforme mencionados acima.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Realizar um estudo sobre os conceitos das tecnologias de geolocalização de pessoas e, a partir disto, realizar uma aplicação de baixo custo, com uso de RFId e arduino.

1.2.2 Objetivos específicos

- Realizar um estudo dos conceitos de geolocalização para ambientes *indoors*;
- Realizar um estudo da tecnologia RFId focando em geolocalização;
- Realizar um estudo da plataforma de prototipagem arduino focando o uso do RFId;
- Aplicar as técnicas estudadas em um estudo de caso dentro do cenário atual do câmpus IFSul Passo Fundo visando à viabilização econômica.

1.3 ORGANIZAÇÃO DA MONOGRAFIA

Este trabalho encontra-se dividido em capítulos, onde o capítulo 2 apresenta os principais conceitos referente as tecnologias abordadas para o desenvolvimento do estudo de caso. O capítulo 3 destina-se ao desenvolvimento do protótipo do estudo de caso, apresentando cada uma de suas fases. No capítulo 4 apresentam-se os testes desenvolvidos com o protótipo e seus resultados. Por fim encontra-se as considerações finais do trabalho, com seus resultados, dificuldades encontradas e trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo encontram-se os principais conceitos abordados, através de pesquisas bibliográficas, baseadas nas principais literaturas da área, compondo o conhecimento necessário para o trabalho.

2.1 GEOLOCALIZAÇÃO

Geolocalização pode ser definida como um sistema que permite a localização da posição geográfica de determinado objeto ou pessoa, através da coleta de dados (MENDES, 2011, p. 25). Tal coleta pode ser feita através de diferentes técnicas, tais como: monitoramento de um dispositivo móvel; uso do GPS; base de dados com endereços IPs ou endereço MAC; sensores; entre outros. Com o avanço da tecnologia, é possível obter informações cada vez mais precisas de localização.

Logo, a posição de um objeto pode ser determinada através de diferentes técnicas de detecção, para diferentes ambientes, pois existem limitações de acordo com o local onde o objeto monitorado encontra-se, seja ambiente interno (*indoor*), como prédios comerciais e residências; ou ambiente externo (*outdoor*), como florestas e cidades.

2.1.1 Geolocalização em ambientes *Outdoors*

Ambientes *outdoors* são caracterizados por terem uma grande área aberta (ampla extensão), facilitando, assim, a identificação da posição do objeto/pessoa. Segundo Garcia (2009, p. 9), esta tecnologia encontra-se mais desenvolvida em relação às de ambientes *indoors*. Para localização de um ponto específico em ambientes extensos, pode-se utilizar infraestrutura baseada em GPS, por se tratar de uma das tecnologias melhor desenvolvida.

GPS:

Pode-se definir o GPS como um sistema de posicionamento para ambientes *outdoors* baseado em satélites.

De acordo com Lima (2001, p. 11):

> A infraestrutura do GPS consiste de 28 satélites em órbita em torno da terra numa altitude de aproximadamente 20.000 Km. Estes satélites carregam relógios atômicos muito precisos. Estes relógios são monitorados continuamente por estações em terra, operados pela Força Aérea norte-americana e possíveis offsets nos relógios do satélite são corrigidos por meio de comandos de controles apropriados partindo destas estações. Cada satélite transmite um fluxo de dados como um sinal RF.

A precisão desta tecnologia é garantida por alguns aspectos determinantes, um exemplo disso, segundo Garcia (2009, p. 4), é que cada satélite possui quatro relógios atômicos, permitindo a marcação do tempo em bilionésimo de segundo, garantindo assim uma precisão maior. Porém, para obtenção da posição, com uma boa precisão, o GPS utiliza de frequências muito altas (GHz), o que requer que exista uma linha reta (sem interferências) entre o satélite e o objeto monitorado.

Os receptores GPS analisam as informações e utilizam técnicas de triangulação (detalhada na próxima sessão) para calcular a posição exata do objeto na superfície terrestre.

2.1.2 Geolocalização em ambientes *Indoors*

Também conhecidos como LPS, os sistemas de localização em ambientes *indoors* são considerados grandes alvos de estudos, pois, conforme Lima (2001, p. 12), "apesar dos avanços extraordinários na tecnologia GPS, milhões de metros quadrados de espaço *indoor* estão fora do alcance de satélites do Navstar[1]".

As soluções para geolocalização *indoor* diferenciam-se das soluções para geolocalização *outdoor* devido à maior necessidade de precisão. Conforme salientado por Lima (2001, p. 13), nos ambientes *outdoors*, dependendo da aplicação, uma boa precisão pode chegar até 30 metros; o que seria inaceitável para aplicações de ambientes *indoors*, onde muitas vezes 10 metros ainda podem ser considerados baixa precisão. Na maioria das vezes, estes sistemas requerem uma precisão de 2 metros ou menos.

Há diversas áreas de atuação onde sistemas de geolocalização *indoor* podem ser aplicados, como: sistemas de estoque; localização de objetos ou pessoas. Além de sua aplicação voltada à localização, pode ser utilizada, também, para armazenamento de informações, as quais são registradas em TAGs e fixadas nos objetos ou seres a qual corresponde.

---

[1] De acordo com Lima (2001, p.12), Navstar é o conjunto de 28 satélites que compõem o GPS.

Técnicas de Detecção de Posição para Ambientes *Indoors:*

Existem diversas técnicas para detecção da posição do objeto. Dentre as principais, destacam-se as técnicas de análise do cenário, proximidade e triangulação. Tais técnicas encontram-se descritas abaixo:

a) Análise do Cenário:

Utiliza de características da cena monitorada, a partir de um ponto qualquer, com a finalidade de obter a conclusão sobre a posição de um objeto.

Ocorre em duas etapas, onde na primeira é realizada uma varredura da área e armazenados os dados referentes à intensidade do sinal em cada um dos pontos. Na segunda, são analisados os dados obtidos com os dados da primeira etapa, sendo, assim, possível estimar a localização do objeto.

Segundo Hightower e Boriello (2001, p. 6), estas análises de cenas podem ser baseadas em imagem visual, como mídias capturadas por uma câmera, ou simplesmente nos fenômenos físicos.

Hightower e Boriello (2001, p. 6) também afirmam que o sistema RADAR[2], desenvolvido pelo grupo de pesquisa da Microsoft, possui duas implementações, onde: uma é desenvolvida por meio de Análise de Cenário e outra por Lateração (Triangulação). Este sistema foi criado para observar um conjunto de dados de medições da intensidade do sinal, através das transmissões de rádio de um dispositivo de rede, observando essas transmissões em várias posições, para assim, determinar a posição de determinado objeto.

b) Triangulação:

Utiliza de propriedades geométricas de triangulação para calcular a posição exata do objeto. É dividida em dois métodos para obtenção da posição, sendo eles:

- **Latência:** Para este método é necessário três antenas, onde é calculada a distância das múltiplas posições das antenas até o objeto.
- **Angulação:** São necessárias duas antenas, baseadas nos ângulos de cada antena em relação ao objeto, determinando para cada uma a direção de onde provém o sinal. Ao cruzar as informações das duas antenas, obtém-se a posição do objeto.

[2] marca registrada de Microsoft Inc.

c) Proximidade:

Para detectar a posição do objeto através de proximidade é preciso que o sistema conheça a posição exata de cada antena, pois ele determina quando o objeto está próximo da posição conhecida. A posição é baseada na intensidade do sinal recebido pela antena. Segundo Hightower e Borrielo (2001, p. 7), para esta técnica existem três abordagens gerais:

- **Detecção física pelo contato com a TAG:** Trata-se do método mais conhecido e utilizado de localização por proximidade. Esta abordagem de localização necessita do uso de um sensor, podendo este ser: sensor por toque, por pressão ou detector por campo capacitivo. Detecções de campo capacitivo estão sendo usado para sistemas de comunicação de dados entre objetos em contato com a pele, por exemplo Monitores *Touch*.
- **Pelo monitoramento dos pontos de acesso:** Monitora quando um dispositivo móvel está no alcance de um ou mais AP na rede *wireless* do celular. Exemplos desta técnica, são o Sistema Xerox ParcTAB, o qual utiliza sensor infravermelho, e o *Carnegie Mellon Wireless Andrew*, o qual utiliza a rede via rádio com protocolo IEEE 802.11.
- **Observação de sistemas de Identificação automática:** Consiste em um sistema de identificação automática, podendo ser utilizadas TAGs de identificação, como em sistemas de portaria eletrônica. Assim, se um sensor realizar a leitura desta TAG, é possível conhecer a localização da mesma.

Ainda de acordo com Hightower e Borrielo (2001, p. 8) a técnica de proximidade, em alguns casos, necessita de combinação da abordagem com um sistema de identificação, quando a abordagem utilizada não fornece a identificação do objeto.

Problemas dos sistemas de localização *indoor:*

Como mencionado nesta sessão, os ambientes *indoors* possuem diversos fatores que dificultam a localização da posição do objeto, fatores estes que precisam ser analisados e superados para o funcionamento do sistema. Os principais fatores são:

- **Propagação dos sinais de radiofrequência:** Em sistemas baseados em radiofrequência, a propagação do sinal cai muito em virtude da reflexão[3].
- **Interferência dos espaços adjacentes *indoor*:** Em ambientes *indoors*, há a delimitação dos espaços (como salas de aula, secretarias, etc.). Muitas vezes os limites

[3] Característica do sinal de rádio, onde obstáculos impedem a propagação do sinal.

físicos de cada espaço confundem-se, e, para isso, é necessária uma engenharia para delimitar tais espaços, a fim de evitar interferências e erros de leitura.

2.2 RFID

A tecnologia RFId pode ser traduzida para Identificação por Radiofrequência. Define-se como um meio de identificar a posição ou a existência de um determinado objeto em determinada área. Segundo Tanenbaum (2011, p. 45), através do RFId, "os objetos do cotidiano também podem fazer parte de uma rede de computadores".

Um dos principais objetivos do RFId é promover uma evolução do código de barras tradicional, por ser um mecanismo de leitura com maior alcance e pela sua capacidade de armazenar mais informações.

2.2.1 Surgimento do RFId

A tecnologia RFId teve sua origem nos sistemas de radares utilizados durante a 2ª guerra mundial, onde eram utilizados para avisar quando aviões se aproximavam. O sistema foi inventado pelo físico Robert Alexandre Watson-Watt, que em sua primeira versão, tinha como objetivo apenas a identificação de aeronaves, porém, na ocasião, o sistema não fazia diferenciação entre aeronaves aliadas ou inimigas.

Após, Watson-Watt aprimorou e desenvolveu um sistema baseado em transmissores, os quais eram inseridos nas aeronaves dos aliados, que passaram a enviar um sinal diferente ao radar, realizando então, a diferenciação entre aeronaves aliadas e inimigas. Logo, surgiu o primeiro sistema de identificação por radiofrequência. A tabela 1 mostra a evolução histórica dessa tecnologia.

Tabela 1 - A evolução do RFId – 2008.

Década	Eventos
1940-1950	Invenção e rápido desenvolvimento do radar durante a 2ª guerra mundial; O princípio de funcionamento do RFId em 1948.
1950-1960	Primeiras explorações e testes laboratoriais do RFId.
1960-1970	Desenvolvimento da teoria RFId; Primeiras aplicações experimentais no terreno.
1970-1980	Crescimento do desenvolvimento de RFId; Acelerações em testes; Implementações embrionárias de RFId.
1980-1990	Aplicações comerciais de RFId entram no mercado.
1990-2000	Surgimento de *Standards;* RFId é largamente utilizado começando a fazer parte da vida cotidiana

Fonte: Adaptado de PRATA, 2008.

2.2.2 Arquitetura básica do sistema RFId

Segundo Passaretti (2008, p. 2), o sistema RFId é baseado em quatro componentes, sendo eles: tags, leitores, *middleware* e *software* de aplicação. Porém, conforme Gines e Tsai (2007), o sistema baseado em sensores RFId é formado por três componentes básicos: transponder, leitor rfid e Computador *Host*. Os componentes identificados são apresentados a seguir:

- ***Transponder*:** Também conhecido por TAG, contém a informação a ser lida. A TAG é fixada no objeto a ser identificado, o qual frequentemente é móvel, possibilitando assim a sua identificação. Detalhado na sessão 2.2.4.
- **Leitor RFId:** É o componente responsável pela leitura das TAGs. Detalhado na sessão 2.2.5.
- **Computador *Host* ou *Software* de Aplicação:** responsável pela coleta, filtragem e encaminhamento dos dados. Detalhado na sessão 2.2.6.
- ***Middleware*:** Interface de aplicação, entre o leitor e o computador *host*, para gerenciamento dos dados. Detalhado na sessão 2.2.7.

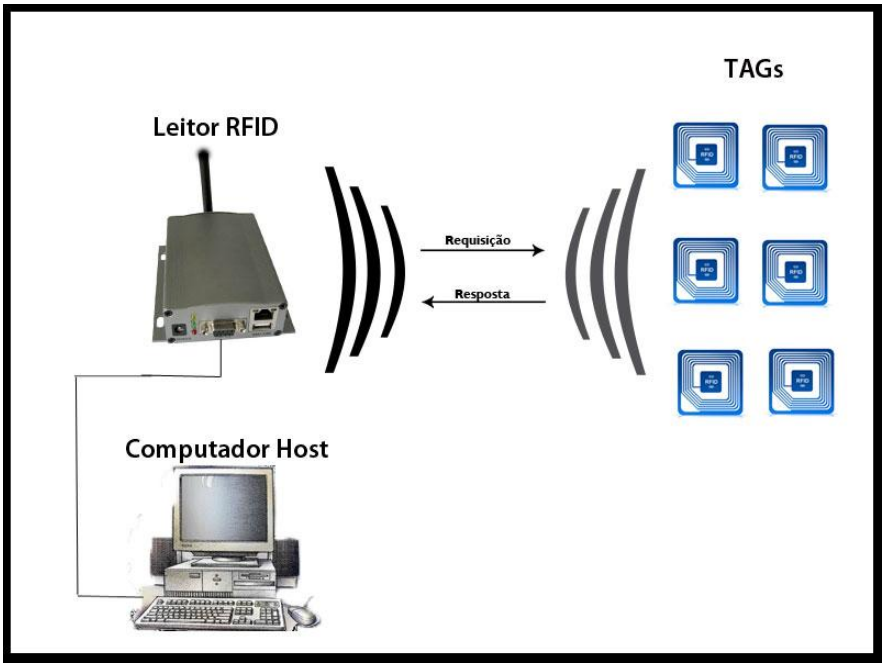
2.2.3 Funcionamento do RFId

Na tecnologia RFId, determinados elementos são “etiquetados” individualmente, onde nestas etiquetas contêm um código de identificação (ID), que é enviado por sinal de radiofrequência do circuito integrado até o dispositivo leitor.

O dispositivo leitor recebe o sinal enviado pela TAG e realiza a identificação desta e o leitor pode, ainda, escrever dados nas TAGs. O decodificador converte este sinal recebido para um código digital, que poderá então ser interpretado pelo computador.

Assim, quando a etiqueta passa por um campo eletromagnético, formado pela antena leitora do RFId, o sinal de identificação é ativado pelo leitor, o leitor decifra os dados contidos nos circuitos integrados da etiqueta e o dado é enviado ao computador *host*. No computador *host*, o *software* faz o processamento dos dados e compara ou atualiza os dados no banco de dados, podendo assim responder à solicitação realizada pela aplicação. A figura 1 demonstra esse procedimento.

Figura 1 - Composição do sistema RFId.



Fonte: Do Autor (Adaptado de imagens retiradas da internet).

2.2.4 TAGs RFId

A TAG, que armazena o ID, é o componente receptor/transmissor do sinal de resposta à solicitação enviada pela estação remota, sendo que, esse retorno pode conter somente o ID ou ainda qualquer dado armazenado nela.

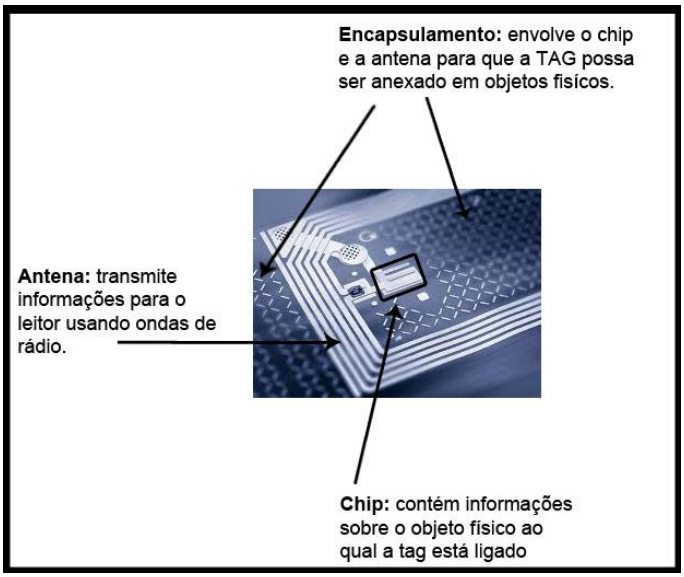
Esta TAG pode ter diversos formatos, que vão desde simples adesivos até cartões inteligentes. Esta diversidade é mostrada na figura 2.

Figura 2 - Exemplos de TAGs RFId comercializadas.

Fonte: Do Autor (Adaptado de imagens retiradas da internet).

Segundo Tanenbaum (2011, p. 45) “uma etiqueta consiste em um pequeno microchip com um identificador exclusivo e uma antena que recebe as transmissões de rádio”. Para Gines e Tsai (2007, p. 10) “a etiqueta é composta por três partes: chip, antena e encapsulamento”, conforme mostrado na figura 3.

Figura 3 - Composição física das TAGs RFId.



Fonte: Composição do Autor (Imagem retirada de ALVES, 2012 apud RABELLO, 2012).

- ***Chip*:** formado por componentes eletrônicos. Sua função é administrar todas as identificações e comunicações realizadas, conferindo sempre que for solicitada uma resposta e se o solicitante é confiável.
- **Antena:** recebe e transmite os dados contidos na TAG. Quando a TAG for do tipo passiva, a antena servirá também para fornecer a energia que a etiqueta necessita para funcionar.
- **Encapsulamento:** material no qual a TAG é embutida para protege-lá.

Tipos de TAGs

Existem algumas diferenças de tecnologia e de tipos de TAGs, que serão fatores determinantes para a escolha das mesmas. De acordo com Gines e Tsai (2007), existem quatro tipos de etiquetas, sendo elas: RFId Passiva, Ativa, Semi-passiva e Semi-ativa. De outro lado, Tanenbaum apresenta a tecnologia, distinguindo-a em dois tipos principais: RFId Ativa e Passiva, as quais são descritas abaixo:

a) RFId Passiva

As etiquetas passivas não possuem fonte de energia própria. São ativadas quando entram no campo de alcance da antena, a qual fornece energia para a TAG através das ondas

de rádio, ou seja, dentro do microchip das TAGs passivas existe um retificador[4] que converte o sinal de radiofrequência (emitido pelo transmissor) em energia elétrica, servindo de alimentação para o chip da etiqueta. Esta conversão de radiofrequência em energia ocorre somente no momento da leitura da TAG.

É definida como uma tecnologia de baixo custo, porém com alcance normalmente menor que das TAGs ativas. O alcance destas é determinado, também, pela frequência[5] em que atuam.

Outra característica das TAGs passivas é que normalmente suas informações são gravadas permanentemente de fábrica, permitindo apenas a identificação do ID armazenado. Esta tecnologia é conhecida como "refletor" de identificação.

No método conhecido como refletor, "[...] As etiquetas se comunicam em distâncias de vários metros, mudando o modo como elas refletem os sinais do leitor." (TANENBAUM, 2011, p. 45). Neste método, a TAG UHF passiva simplesmente reflete as ondas enviadas pelo leitor, onde a energia do leitor é captada pela TAG como fonte de energia, armazena brevemente e transmite de volta para o leitor. Porém, este método pode resultar em colisões quando mais de uma TAG reflete a energia de volta ao leitor simultaneamente, confundindo o leitor. O modo de operação HF utiliza o mecanismo conhecido por indução, que de acordo com Want (2006 apud Souza, Silva e Barroso 2010, p. 22):

> A tecnologia RFID usa este princípio para gerar energia para os circuitos eletrônicos presentes em um tipo de etiqueta. Ao utilizar um campo magnético variável próximo a uma etiqueta, é produzida em uma bobina interna da etiqueta uma corrente elétrica induzida, que alimentará os circuitos do microcontrolador e transmissor de dados. A corrente gerada deve possuir intensidade suficiente para alimentar esses circuitos. O leitor RFID deve gerar um campo elétrico variável de intensidade suficiente para garantir a potência necessária para a alimentação da etiqueta.

b) RFId Ativa

Diferente das TAGs passivas, estas possuem uma fonte de energia própria e em sua maioria possuem um radiotransmissor. Com isso, possibilita a transmissão de dados remotamente, pois possui a habilidade de iniciar a comunicação com o leitor e também de transmitir os dados a distâncias maiores quando comparadas com as passivas.

Porém, devido ao uso de bateria, que tem vida útil relativamente pequena, o custo desta tecnologia, que já é superior, torna-se ainda mais cara.

---

[4] Dispositivo utilizado para converter corrente alternada em corrente contínua e alimentar a TAG.

[5] Descrito nesta sessão, item Frequência de operação da TAG.

Frequência de operação da TAG

De acordo com Dias e Baladei (2012) “a frequência define a taxa de transferência de dados entre a etiqueta e o leitor, mas a velocidade não é o único aspecto a ser analisado em uma solução RFId”.

O alcance para leitura de uma TAG em relação à antena está diretamente relacionado com a frequência em que se está operando. Existem ainda, outros aspectos relacionados que podem influenciar no alcance de uma TAG, como: dimensão da antena, posição desta em relação à etiqueta, ou mesmo, o ambiente em que está operando.

Por exemplo, ambientes com alta concentração de metais podem interferir nas ondas de rádio, neste caso equipamentos que operam em baixa frequência, têm maior facilidade de adequação neste tipo de ambiente sem sofrer interferência. Isso pode ser verificado na tabela 2, que traz a relação entre a frequência das TAGs RFId e seu respectivo alcance, baseado em adaptações realizadas sobre a tabela: Características das diferentes frequências de rótulos RFId, de Gines e Tsai (2007, p. 14).

Tabela 2 - Características das diferentes frequências das TAGs RFId.

Frequência	**Alcance de leitura**	**Capacidade de leitura em ambientes diversos (metais e líquidos)**	**Dimensão da etiqueta**	**Fonte de energia**	**Comentário (Aplicação usual)**
Baixa Frequência (LF) 125 - 134,2 KHz	< 0,5 m	Excelente	Muito Grande	TAG passiva, fonte energia resultante da indução eletromagnética	Animais tagging e controle de acesso
Alta Frequência (HF) 13.56MHz	< 1,0 m	Boa	Grande	TAG passiva, fonte energia resultante da indução eletromagnética	*Smart Card* e controle de acesso
Frequência Ultra Alta (UHF) 860 - 960 MHz	> 3,0m	Média	Média	Ativas: possuem energia própria (bateria), ou passiva.	*Containers* tagging, sistema de transporte
Microondas 2,45GHz e 5,8 GHz	5,0 - 10,0m	Baixa	Pequena	Ativas: possuem energia própria (bateria), ou passiva.	Cadeia de suprimentos, sistema de transporte

Fonte: Adaptado de GINES e TSAI, 2007.

Devido a menor influência dos metais sobre as TAGs em LF, estas são utilizadas em aplicações em que a TAG fica incorporada ao metal, como por exemplo, em instrumentos cirúrgicos. As TAGs HF, de alcance menor que um metro, são mais adequadas ao controle de acesso; com TAGs incorporadas em vastas opções de objetos, como chaveiro e cartões, e por demandar de um alcance restrito e médio, protege o sistema de leituras indevidas.

Padronização das TAGs para o RFId

A Padronização da codificação das TAGs é um item de suma importância na aplicação dessa tecnologia, pois até sua origem, os fabricantes utilizavam sistemas proprietários para essa função, gerando uma diversidade de protocolos e tornando inviável a comercialização do sistema RFId. Atualmente existem duas tecnologias de protocolos em vasta utilização: ISO e EPC.

a) ISO

A ISO criou muitos padrões de RFId que lidam com o protocolo da interface aérea e aplicações para RFId, responsável pelos protocolos em RFId a mais de 20 anos. Segundo a *Nationals Instruments* (2011), durante muito tempo prevaleceu como principal órgão padronizador de protocolos, mesmo com a criação do EPC, pois vendedores apoiavam os protocolos ISO por se tratar de uma tecnologia já desenvolvida. A tabela 3 mostra alguns padrões publicados pela ISO para TAGs passivas e seu avanço até 2004.

Tabela 3 - Relação das padronizações para TAGs Passivas.

ISO Standard	Título
ISO 11784	RFId para animais – Define a estrutura de dados da marcação de animais
ISO 11785	RFId para animais – Concepção técnica da comunicação
ISO/IEC 14443A, B	Identificação de cartões – cartões com circuitos integrados sem contato – cartões de proximidade. Padrão utilizado quando o alcance de leitura é limitado em menos de 10cm.
ISO/IEC 15693	Identificação de cartões – cartões com circuitos integrados sem contato – cartões de vizinhança. Publicado em 2000, normalmente utilizado para leitura com alcance de mais de 10cm.
ISO/IEC 18001	Tecnologia da Informação – Gerenciamento de Itens de RFId – Perfil de Requisitos de Aplicação
ISO/IEC 18000-1	Parâmetros Gerais para Comunicação por Interface, para Freqüências Globalmente Aceitas
ISO/IEC 18000-2	Parâmetros para Comunicação por Interface, para LF (abaixo de 135 KHz). Publicada e finalizada em 2004.
ISO/IEC 18000-3	Parâmetros para Comunicação por Interface e trabalha na freqüência 13.56 MHz. Publicada também em 2004.
ISO/IEC 18000-4	Parâmetros para Comunicação por Interface por Ar em 2.45 GHz
ISO/IEC 18000-6	Parâmetros para Comunicação por Interface, para UHF (860 a 956 MHz)
ISO/IEC 18000-7	Parâmetros para Comunicação por Interface, para freqüência 433 MHz
ISO/IEC 15961	Gerenciamento de Itens de RFId – Protocolo de Dados: Interface de Aplicação
ISO/IEC 15962	Gerenciamento de Itens de RFId – Protocolo: Regras de Codificação de Dados e Funções de Memória Lógica
ISO/IEC 15963	Gerenciamento de Itens de RFId – Identificação única do RF Tag

Fonte: Adaptações de *Understanding Passive RFId Tecnology* – RFID *World.*

Porém, após a publicação destes protocolos houve muitos avanços e alguns protocolos definidos pela ISO que prevalecem até os dias de hoje, como: para UHF o padrão 18000-6C (EPCGlobal Gen2), e para a faixa de 433MHz, em TAGs ativas, o padrão ISO 18000-7, para HF os padrões ISO 15693 e ISO 14443.

De acordo com a empresa Atmel, o protocolo ISO/IEC 14443, opera na frequência 13.56 MHz e seu alcance de leitura em relação ao leitor é de até 10 cm, além de ser dividido

em dois tipos: tipo A e tipo B. Constitui-se das seguintes partes: características físicas: que abrangem a identificação de cartões, distância para leitura dos cartões e cartões de circuito integrado sem contato; sinal para operação por radiofrequência: para esquema de comunicação, utiliza o modo *half duplex* com uma taxa de 106 Kbit/s de dados em cada direção, os dados transmitidos pelas TAGs são carregados no módulo com uma frequência da subportadora de operação de 847,5 KHz e ,além disso, o cartão é alimentado pelo campo de radiofrequência, não sendo necessário o uso de bateria; e protocolo anticolisão: que define a inicialização e o protocolo anticolisão para os tipos A e B.

b) EPC

O EPC é um padrão criado pelo órgão regulador internacional EPCglobal e, no Brasil, este órgão é representado pela GS1[6], responsável pela padronização contida nas TAGs. Surgiu originalmente como um protocolo para substituição do código de barras e, segundo a *National Instruments* (2011), "O EPC foi fundado por usuários finais como o Wal-Mart, Gillete e PNG e eles buscavam uma tag barata e um protocolo que a suportasse". Além disso, na criação deste protocolo, tinha-se como lema a seguinte afirmação: "Torne isso o mais simples possível para que o chip possa ser o mais simples possível e o custo da TAG possa ser o mais baixo possível".

Em julho de 2006, os órgãos padronizadores ISO e EPC uniram-se na criação de um novo protocolo: EPCGlobal Gen2, adotado pela ISO como padrão ISO 18000-6C. O EPCGlobal Gen2, é o protocolo mais avançado para UHF, pois sua criação foi baseada nas melhores práticas dos protocolos EPC Gen1 e ISO 18000-6. Dentre suas vantagens, esse protocolo trabalha com sessões, onde a TAG possui quatro sessões, com isso é possível que quatro leitores se comuniquem com a TAG, simultaneamente, sem que haja conflito. (*NATIONAL INSTRUMENTS*, 2011).

De acordo com o RFID *Journal* Brasil (2003), os códigos EPC são compostos de um cabeçalho e três conjuntos de números, como no exemplo "1-2345-67890", onde o primeiro conjunto representa a identificação do fabricante, o segundo a identificação do tipo do produto e o último representa o código de identificação único da TAG. Assim, é possível realizar a programação do leitor, a fim de separar os conjuntos e realizar as buscas que forem convenientes.

---

[6] GS1 - Associação Brasileira de Automação: http://www.gs1br.org/

2.2.5 Leitor RFId

Trata-se do componente intermediário para o funcionamento da tecnologia, pois realiza o interfaceamento entre o sistema solicitante e os dados fornecidos pelas TAGs. Funciona através de radiofrequência, não necessitando de contato visual com a TAG; além disso, pode realizar a leitura de várias TAGs simultaneamente. Além dos comandos para leitura dos dados, o leitor também pode realizar a escrita na etiqueta, se essa for compatível.

"A tarefa principal da leitora é fazer o inventário das etiquetas nas vizinhanças, ou seja, descobrir os identificadores das etiquetas vizinhas" (TANENBAUM, 2011, p. 206). Esses equipamentos "[...] possuem sua própria fonte de energia, capacidade de processamento e uma antena para comunicação" (MOTA, 2012, p. 6) e, também, podem possuir várias antenas e definir quando as etiquetas devem enviar ou receber mensagens.

O leitor é responsável pela conversão dos dados recebidos para uma linguagem digital, para que possa ser entendida pelo computador solicitante. A figura 4 mostra alguns formatos e modelos de leitores.

Figura 4 - Modelos e Formatos de alguns leitores RFId.

Fonte: Do Autor (Adaptado de imagens retiradas da internet).

2.2.6 Computador *Host*

Trata-se do computador ou servidor responsável por receber as informações do leitor e repassar ao sistema solicitante da informação. Esse computador poderá conter um *middleware*.

2.2.7 *Middleware*

O *middleware* trata-se do componente, ou *software*, responsável pela conversão das informações obtidas pelo leitor, as quais são enviadas ao computador *host*, com intuito de mostrar os dados obtidos à aplicação solicitante.

Para Loureiro et al. (2003, p. 221), o *middleware* "permite que informações adquiridas no canal de sensoriamento sejam repassadas para a pilha de protocolos de rede, a fim de serem transmitidos a outro nó". Tais fatos são também confirmados por Zanlourensi (2011), ao mencionar que "ele é o responsável pela depuração das informações recebidas pelas antenas (eliminando redundâncias, etc) e conversão dessas informações em algo que o sistema do usuário possa interpretar".

> *Middleware* RFId é um *software* para o sistema de identificação por radiofrequência que intermedia a comunicação entre o sistema da organização e a infraestrutura de *hardware* do sistema RFId, formada por leitores e etiquetas ou sensores que estão acoplados à esta rede. (DIAS, 2012).

Dias (2012), ainda coloca que nem sempre o sistema RFId exige o uso do *middleware*, pois seu uso é necessário sempre que o número de dados coletados é alto, visto que, além de suas funções usuais, ele pode monitorar e gerenciar os *hardwares* que compõem o sistema.

2.2.8 Regulamentação referente à Frequência

No Brasil, a ANATEL estabelece as condições de uso de radiofrequência dos equipamentos que utilizam o espectro de frequência para a transmissão de um sinal. De acordo com a Resolução nº 506 (de 1º de julho de 2008), a qual republica o regulamento sobre equipamentos de radiocomunicação de radiação restrita, em seu artigo 2, inciso XIV, traz o conceito de RFId como: sistema, composto por dispositivo transceptor, que recebe e envia

sinais de radiofrequências, quando excitado por um equipamento transceptor interrogador, que tem a capacidade de efetuar a leitura, escrita ou modificação das informações contidas no dispositivo.

A seção XII (Sistemas de Identificação por Radiofrequências), Art. 52, define que devem operar nas frequências 119-135 kHz, 13,11-13.36 MHz, 13.41-14.01 MHz, 433.5-434.5 MHz, 860- 869 MHz, 894-898.5 MHz, 902-907.5 MHz, 915-928 MHz, 2400- 2483.5 MHz e 5725-5850 MHz e devem atender aos limites de intensidade de campo elétrico, conforme disposto na resolução.

2.3 ARDUINO

Define-se Arduino como uma tecnologia baseada em um microcontrolador, com propósito de detectar e controlar *hardware* e *software*. Possui um ambiente que permite a escrita de códigos para a programação da placa. Sua tecnologia é *open-source*[7], o que permite a distribuição dos códigos para utilização, modificação e redistribuição por qualquer pessoa (é *open-source* tanto para *software*, quanto para *hardware*).

Segundo Arduino (2013, tradução nossa), o arduino é uma plataforma de prototipagem eletrônica, *open-source* baseado em *hardware* e *software*, flexível e fácil de usar.

O arduino surgiu para desenvolver a interatividade entre objetos, sendo que pode ser programado para processar entradas e saídas de dispositivos e componentes externos conectados a ele através de s*hields*, que são placas que podem ser conectadas ao arduino a fim de obter funcionalidades adicionais, ou também, através de módulos contendo a função necessária.

Segundo Monk (p. xi, tradução nossa),

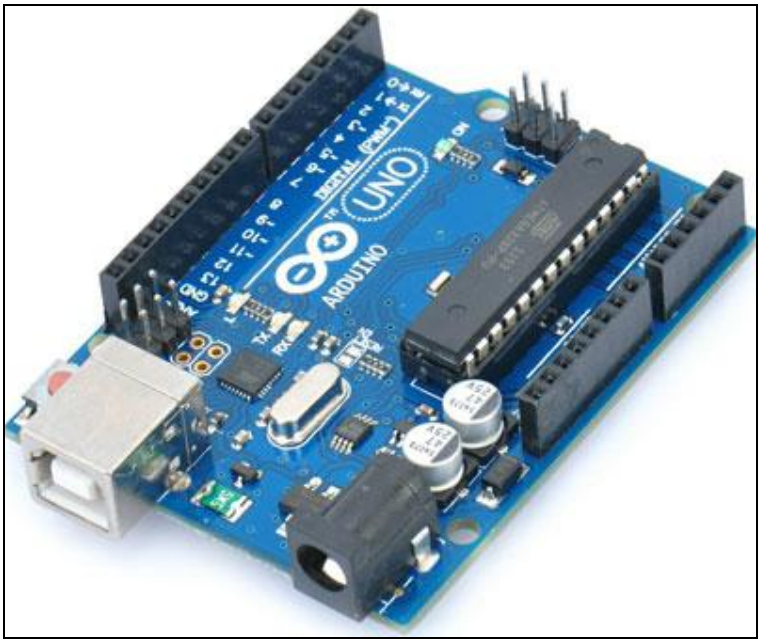
> Arduino é uma pequena placa de microcontrolador com um plug USB para se conectar ao computador e um número de conectores para a ligação, onde podem ser conectados componentes eletrônicos externos, tais como: motores, sensores de luz, diodos de laser, alto-falantes, microfones, etc.

[7] Criado pela Open Source Iniciative (OSI), determina que um programa de código aberto deve garantir a sua distribuição livre, código fonte, integridade do autor do código fonte e, distribuição de licença.

2.3.1 *Hardware*: Componentes Físicos do Arduino

De acordo com Mcroberts (2011, p. 23) “A placa do arduino é composta de um microprocessador Atmel AVR, um cristal ou oscilador (relógio simples que envia pulsos de tempo em uma frequência especificada, para permitir sua operação na velocidade correta) e um regulador linear de 5 volts”, tal placa encontra-se ilustrada na figura 5. Em sua maioria conectam-se ao computador através de uma saída USB. Além de sua estrutura básica, a placa do arduino pode ser complementada com uma variedade de módulos adicionais, através do uso de *shields* ou módulos. Cada placa possui, também, uma memória de execução e outra memória flash.

Figura 5 - Arduino UNO.



Fonte: Arduino.

Segundo Tanenbaum (2007 apud Borges, 2012, p. 25), “microcontroladores são considerados computadores pequenos, mas completos, compostos por processador, memória e capacidade de E/S os quais permitem detectar botões e interruptores do aparelho além de controlar suas luzes, monitores, sons e motores”.

A alimentação do arduino pode ser fornecida pela conexão USB ou mesmo por uma fonte externa, tais como baterias e pilhas. (PRADO, 2012, p. 20).

Entretanto, para que haja interação com os componentes é necessário que a placa microcontroladora contenha uma programação, trata-se de uma IDE para geração dos programas que serão enviados a placa do arduino. Conforme Arduino (2013, tradução nossa), sua IDE é um compilador avr-g++ da linguagem de programação C/C++, a qual é escrita na linguagem de programação Java e sua plataforma é suportada pelos sistemas operacionais Windows, Linux e Mac OS X.

2.3.2 RFId com Arduino

O arduino pode ser utilizado como um automatizador das funções do RFId, onde para isso, é necessário utilizar um módulo que adicione esta função à placa. Diversos módulos estão disponíveis para uso com o arduino. Cada tipo de módulo possui operações e comandos próprios, definidos por cada fabricante. Assim, os comandos para realização das operações também são específicos para cada módulo, dentre estes comandos está a leitura do serial contido nas TAGs, tornando assim, possível o envio do comando ao arduino, e se houver TAGs ao alcance do leitor, este enviará como resposta o serial da TAG.

A estrutura básica deste sistema é relativamente simples, onde a leitura de uma TAG ocorre pelo leitor RFId, função esta disponível pela adição do módulo, a qual envia os dados ao arduino, e daí então, as ações pré-programadas são executadas.

3 DESENVOLVIMENTO DO PROTÓTIPO

Este capítulo apresenta o cenário proposto para o estudo de caso, bem como o desenvolvimento de cada uma das fases para implementação do protótipo, desde a estrutura física e lógica até a visualização dos dados em uma aplicação final.

3.1 PROJETO

O presente projeto visa desenvolver uma aplicação de baixo custo para identificação da posição de determinada pessoa em um determinado espaço fechado (*indoor*), como ambientes empresariais ou residenciais. Para que o sistema seja eficiente, a posição identificada da pessoa deve possuir alta precisão.

A fim de tornar o projeto viável, onde os requisitos propostos fossem possíveis, verificou-se a necessidade de utilizar um microcontrolador para suporte a tecnologia adotada, assim como a utilização de um sensor para identificação da posição do indivíduo, além das aplicações necessárias.

Inicialmente foi realizado um estudo teórico, através de pesquisa bibliográfica, sobre geolocalização aplicada a ambientes *indoors* e sobre a tecnologia RFId, objetivando definir e implementar um estudo de caso. Este estudo embasou algumas decisões na aplicação, tais como quais os tipos de TAG RFId apropriados para este estudo de caso ou com quais mecanismos de controle seriam feitas as leituras dos sinais.

Conforme verificado na revisão bibliográfica, sessão 2.2.2, em arquitetura básica do sistema RFId, o sistema necessita de: TAGs, Leitor, Computador *Host* e *Middleware*. Assim, para a elaboração do projeto, optou-se pela divisão do sistema em camadas, que são representadas na figura 6 e detalhadas a seguir.

Figura 6 - Arquitetura das camadas do projeto de sistema de geolocalização *indoor.*

Fonte: Do Autor.

a) Camada Física: é a camada de mais baixo nível, onde se encontra o *hardware* do projeto, como: microcontrolador, módulo de leitura RFId e componentes eletrônicos. Além disso, também é responsabilidade desta camada o *firmware*[8] do projeto;

b) Camada Intermediária: está entre a camada usuário e a camada física; faz a interação dos componentes físicos com o computador *host*, permitindo assim, que os dados captados pelos sensores sejam enviados ao servidor. Vale salientar que esta camada é "invisível" aos usuários do sistema. Além disso, está presente nesta camada o banco de dados utilizado no projeto; e

c) Camada Usuário: é responsável pela apresentação da aplicação do sistema, bem como a interação necessária entre o usuário final e a aplicação desenvolvida, através de um terminal.

3.2 ESTUDO DE CASO

Atualmente, localizar uma pessoa em um ambiente fechado, quando este possui uma área consideravelmente ampla, não é uma tarefa fácil, podendo se tornar uma tarefa exaustiva. No IFSul - câmpus Passo Fundo, por exemplo, no cenário atual, a localização de um professor é extremamente ineficiente, pois trata-se de uma tarefa onde é preciso ir até a recepção para ver se o professor retirou alguma chave, e então ir até esta sala para ver se o mesmo se encontra nela; caso contrário, é necessário dirigir-se a sala dos professores, e, por vezes,

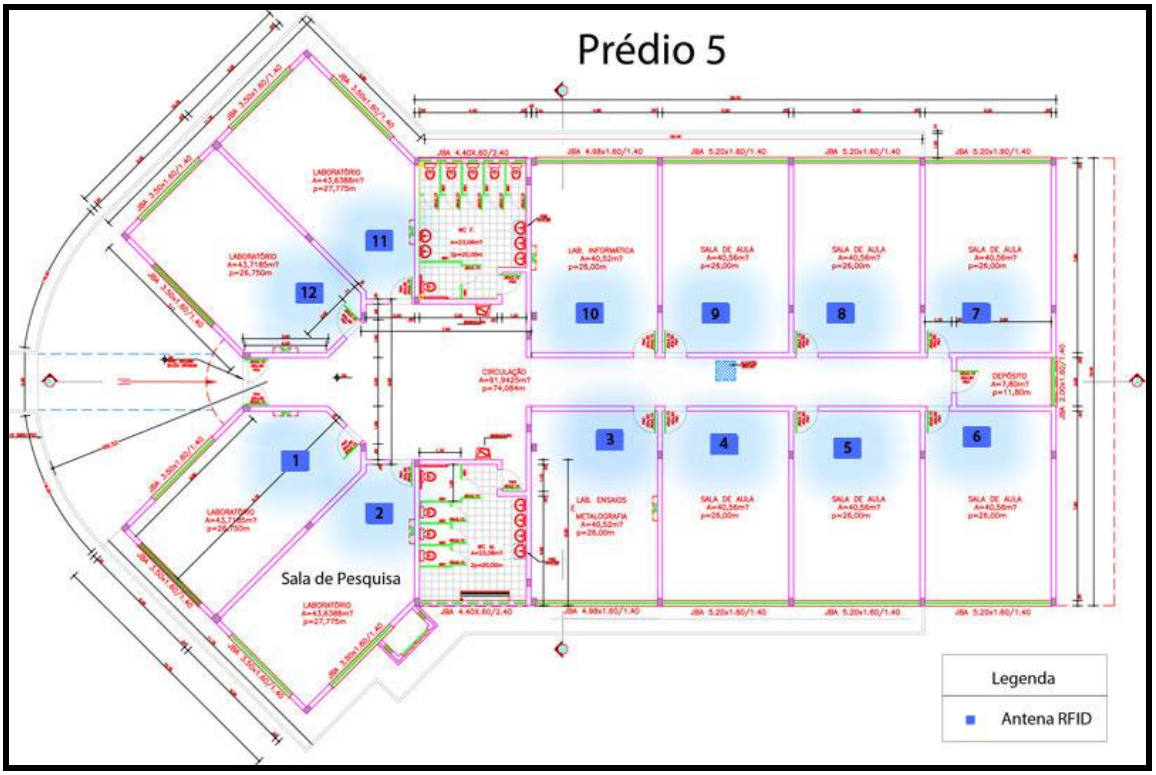
[8] *Firmware* é o conjunto de instruções operacionais programadas diretamente no *hardware* de um equipamento eletrônico.

voltar a recepção. Logo, a procura dos professores, sem uma localização prévia bem definida, torna o processo complicado e causa muitos transtornos, além da perda de tempo. Desta forma, neste estudo de caso, busca-se identificar o posicionamento dos professores, no câmpus Passo Fundo, através de um terminal e uma aplicação. Evidentemente, o sistema pode ser facilmente expandido para a localização de servidores e/ou colaboradores do câmpus.

Para isso é necessário a implantação de um microcontrolador arduino, um leitor RFId, bem como o devido circuito para seu funcionamento. Esta estrutura física deverá estar acoplada nas salas de aula, a fim de identificar o posicionamento dos professores dentro do câmpus. Na figura 7 são apresentados os pontos propostos para instalação das antenas RFId nas salas de aula do IFSul, porém a figura ilustra os pontos em um determinado prédio, sendo que para a implantação completa desta solução, é necessário a colocação dos sensores em todas as salas e em todos os prédios.

Entretanto, com o intuito de validar o protótipo, e também devido aos custos de cada kit e módulo RFId (verificáveis no item Recursos da sessão 3.3.1), o monitoramento inicial foi feito somente em uma das salas de aula.

Figura 7 - Área de Cobertura proposta.

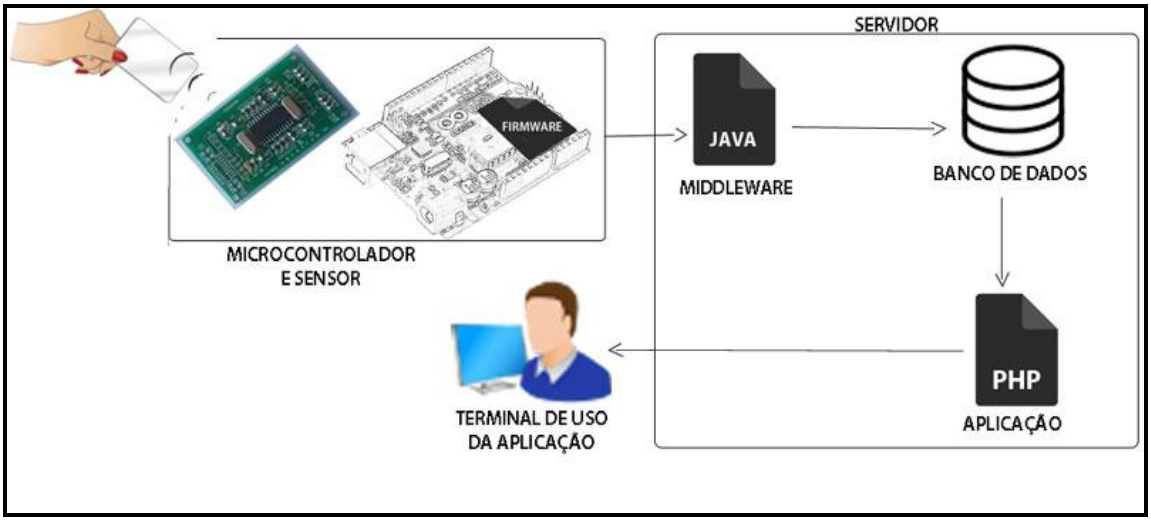


Fonte: Do Autor (adaptações das plantas do IFSul).

3.3 ARQUITETURA DO SISTEMA

A arquitetura proposta para o desenvolvimento do estudo de caso deriva da arquitetura das camadas do projeto de sistema de geolocalização *indoor*, proposto na figura 8. Assim, a figura 8 demonstra a modelagem que foi utilizada para desenvolvimento do protótipo, tal como a elaboração do presente projeto.

Figura 8 - Composição do Estudo de Caso.



Fonte: Do autor.

De forma genérica, pode-se traduzir esta arquitetura como: a camada física do projeto (microcontrolador e sensor RFId), a qual possui um controlador programado na linguagem de programação C++, se conecta ao computador através da programação desenvolvida em Java, conhecido como o *middleware* do sistema, onde realiza a conexão da parte física do sistema com a parte lógica, recebendo os dados captados pelo sensor e conforme os dados são captados pelo sistema são enviados ao banco de dados. A aplicação do sistema recupera os dados do banco de dados, realizando, assim, a integração entre as camadas intermediária e a de interação com usuário. Os detalhamentos destas camadas, bem como os serviços contidos em cada uma, estão descritos nas próximas sessões.

3.3.1 Camada Física

As ferramentas e materiais utilizados na camada física do projeto, assim como os serviços e os itens, compõem esta camada e encontram-se descritos a seguir.

Estrutura Física

As pesquisas realizadas na revisão bibliográfica demonstraram que, utilizar leitor e TAGs na frequência de trabalho HF (13.56 MHz) seria suficiente para aplicação em ambientes fechados, pois possui um alcance de até um metro de distância e os equipamentos UHF até três metros. Como o cenário do estudo de caso envolve salas de aula, alocadas lado a lado, o leitor RFId deveria ser fixado próximo à porta de entrada, onde uma distância de até três metros poderia gerar erros de localização, ou seja, portas muito próximas poderiam registrar a identificação da mesma pessoa em duas salas ao mesmo tempo, o que resultaria em erros nos dados apresentados na aplicação final. Além disso, a aquisição de leitores UHF tornaria o projeto economicamente inviável.

Sendo assim, após a verificação dos requisitos que o leitor RFId deveria atender, optou-se pela aquisição do módulo YHY502CTG, o qual poderia ser controlado pelo microcontrolador arduino e atende aos requisitos pré-estabelecidos. Este utiliza a técnica de aproximação através do método observação de sistemas de identificação automática. Por se tratar de um módulo que opera na frequência 13.56 MHz, a leitura das TAGs é realizada utilizando o método de indução eletromagnética. Todavia, na prática, o alcance máximo com o módulo leitor YHY502CTG foi muito pequeno (aproximadamente 5cm), resultando em uma alteração no sistema proposto. Devido a esse curto alcance, o professor terá que se identificar ao entrar na sala de aula. Esta alteração no sistema resultou também em uma solução ao problema de interferência dos espaços adjacentes *indoor*.

Em relação às TAGs utilizadas, optou-se pelas TAGs passivas, visto que há necessidade de curto alcance de leitura da TAG, além de não precisarem do uso de baterias, evitando, assim, a necessidade de substituição das mesmas com uma grande frequência que inviabilizasse o projeto. Para isso, optou-se pelas seguintes TAGs: Cartão RFId 13.56 MHz MIFARE 1Kb, o qual utiliza o padrão ISO14443A, e Chaveiro RFId 13.56 MHz MIFARE 1Kb (conforme ilustradas na figura 9). A aquisição de mais de uma TAG se deu em virtude da possibilidade de simulação de mais de um professor no sistema.

Figura 9 - Tags utilizadas no projeto.

Fonte: Do autor.

- Módulo YHY502CTG

É uma placa externa de controle RFId, com funcionamento na frequência 13.56 MHz, compatível com TAGs passivas e possui capacidade de leitura e escrita. Ele é integrado com nove pinos, conforme disposto na tabela 4.

Tabela 4 - Informações Pinos J1 Módulo RFID.

Pino	Símbolo	Descrição
J1-1	RXD	Uart Receiver
J1-2	TXD	Uart Transmitter
J1-3	OUT_1	Saída 1
J1-4	OUT_2	Saída 2
J1-5	RST	Reset
J1-6	BUZ	Conexão do driver buzzer
J1-7	SIG	
J1-8	VCC	Power Positive
J1-9	GND	Power Negative

Fonte: Adaptações de J1 Pin Information – Datasheet YHY502CTG, 2010.

- Quadro de dados do Módulo:

O envio de comandos ao módulo segue o seguinte quadro de dados:
[Cabeçalho + Tamanho da Mensagem + Comando + Dados + *Checksum*]

A resposta do módulo em relação ao comando enviado segue o seguinte quadro de dados:

[Cabeçalho + Tamanho da Mensagem + *Status* + Resposta + *Checksum*]

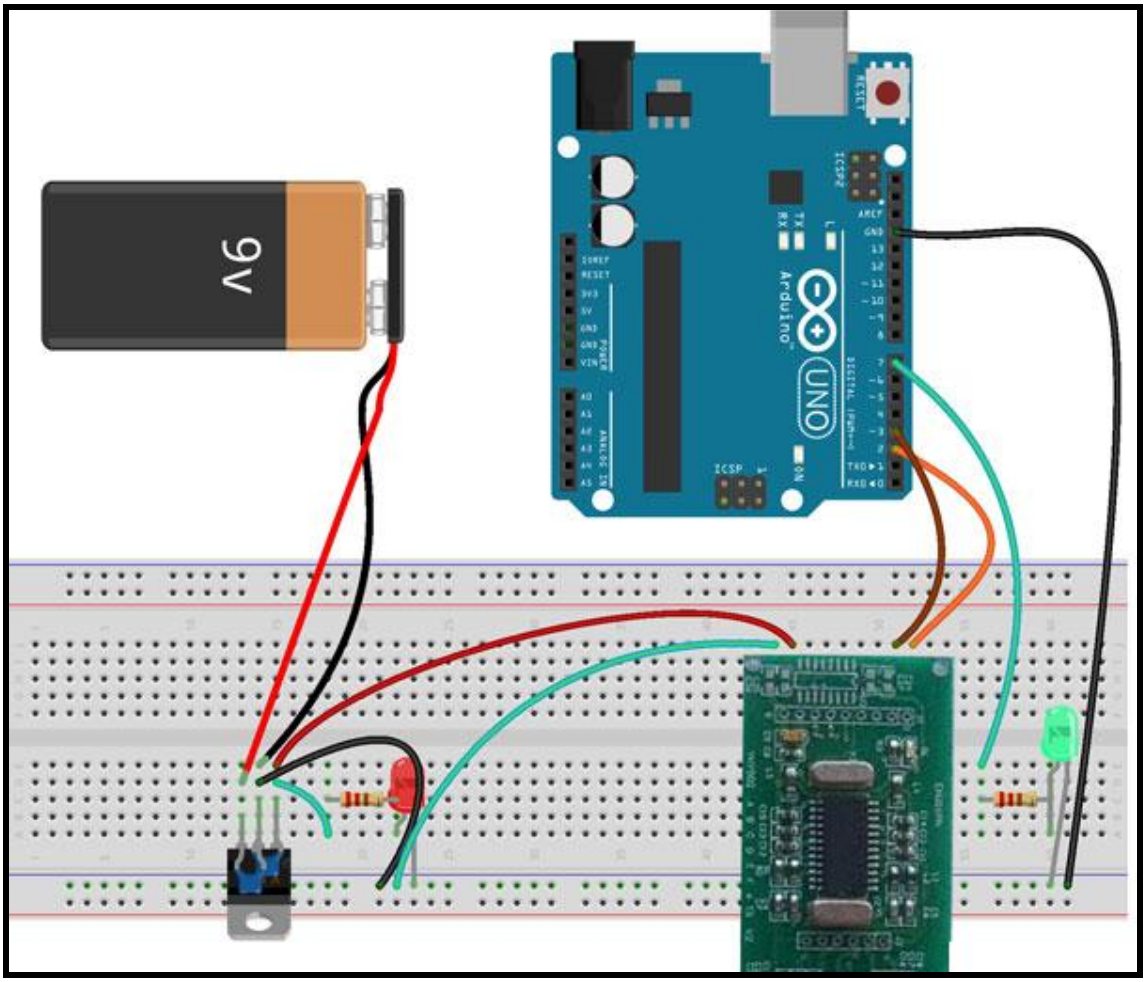
Onde:

- Cabeçalho: Este cabeçalho é composto por 2 *bytes*. Os dados enviados nesse pacote são sempre 0xAA e 0xBB.
- Tamanho da mensagem: Esse pacote indica o tamanho dos pacotes enviados, o qual inclui o tamanho da mensagem, tamanho e os dados.
- Comando: utilizado para instruir o módulo sobre a operação que o mesmo deve executar.
- Dados: Trata-se dos parâmetros para execução do comando, o qual pode ou não ser vazio.
- *Checksum*: Este *byte* é o cálculo dos pacotes enviados e é usado no *host* para checar a integridade dos dados. Seu cálculo é realizado somando todos os *bytes* que compõem o quadro de dados, exceto os *bytes* correspondentes ao cabeçalho e o próprio *Checksum.*
- *Status*: Representa o *status* para o qual a resposta é enviada como retorno ao comando enviado. Se a execução do comando enviado ao módulo estiver ok, o retorno do comando é enviado pelo módulo, caso contrário retorna um complemento do código enviado.
- Resposta: Trata-se dos dados do resultado se a operação foi bem sucedida, caso contrário o campo pode retornar vazio.

Os comandos aceitos pelo módulo são utilizados conforme lista estabelecida pelo fabricante, porém para o presente projeto foi utilizado somente o comando para leitura do *serial number* das TAGs, que é 0x20. Dessa forma, para realizar o controle do módulo, considerando a necessidade de leitura das TAGs, foi enviado ao módulo o comando de leitura do *serial number* contido na TAG, composto pelo seguinte quadro de dados: [0xAA, 0xBB, 0x02, 0x20, 0x22]. Obtendo como resposta o seguinte: [0xAA, 0xBB, 0x06, 0x20, número do serial contido na TAG, *Checksum*]. O número do serial contido na TAG é composto por quatro *bytes* com retorno em hexadecimal.

O módulo YHY502CTG opera entre 3,3v e 5v e, para sua alimentação elétrica, foi utilizada uma bateria de 9v ligada em regulador CI 7805 que tem, na saída, uma tensão fixa de 5v. O circuito utilizado foi elaborado conforme ilustrado na figura 10.

Figura 10 - Esquema do Circuito utilizado para prototipação.



Fonte: Do autor.

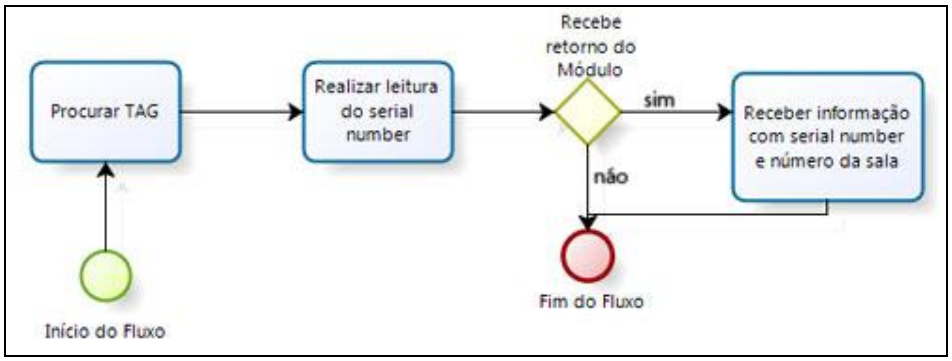
Firmware

Firmware é o *software* de controle da estrutura física, ou seja, do microcontrolador e do módulo RFId; possibilita que os componentes físicos tragam os resultados esperados. O *firmware* compõe parte do *middleware* proposto.

O controlador físico foi construído com a seguinte lógica: verifica se existe alguma TAG no alcance e, se tiver, é enviado o comando para leitura do serial da TAG, se o módulo retornar uma resposta de sucesso, então o código lido é enviado ao monitor serial concatenado com o número da sala onde o módulo encontra-se localizado, o trecho de código está ilustrado

no quadro 1. Em outras palavras, o arduino envia pela serial uma string composta da seguinte forma: [SERIAL_TAG_PROFESSOR;NUMERO_SALA], possibilitando, desta forma, a recuperação dos dados pelo *middleware* na camada intermediária. Este fluxo de dados encontra-se disposto conforme figura 11.

Figura 11 - Fluxo dos dados no *Firmware.*



Fonte: Do autor.

A linguagem de programação utilizada na codificação do *firmware* foi C++, e o ambiente de programação foi a IDE do próprio arduino.

Quadro 1 - Construção lógica do controlador.

```
1.  void loop(){
2.    if (searchCard() != -1) {
3.    digitalWrite(ledPin, HIGH);
4.      for(i=0; i<4; i++){
5.        Serial.print(searchRES[i],HEX);
6.        Serial.print("|");
7.      }
8.    Serial.print(";");
9.    Serial.print(SALA,DEC);
10.   Serial.print("\n");
11.   delay(1000);
12.   digitalWrite(ledPin, LOW);
13.   delay(1000);
14.   }
15. }
```

Fonte: Do autor.

- Biblioteca para controle do módulo

O módulo RFId YHY502CTG utiliza comunicação serial para envio e recebimento de dados, porém o arduino permite leitura serial apenas nas portas 0 e 1 da placa microcontroladora. Esse suporte serial nativo ocorre através de um componente de *hardware*

chamado UART, que controla a comunicação serial, mesmo enquanto trabalha em outras tarefas. Por isso, foi necessário o uso da biblioteca “SerialSoftware.h”, que permite ler serial em qualquer pino.

Esta biblioteca permite comunicação serial em outros pinos, utilizando o *software* para replicar a funcionalidade, a utilização dela está ilustrada no quadro 2. Assim, é possível ter vários pinos seriais com velocidade de até 115.200 bps.

Quadro 2 - Utilização da bilbioteca SoftwareSerial.h.

```
1. #include <SoftwareSerial.h>
2. SoftwareSerial rfid(2, 3);
3. void setup()
4. {
5.     Serial.begin(9600);
6.     rfid.begin(19200);
7. }
```

Fonte: Do autor.

- Desenvolvimento do *Firmware*

O arduino utilizado foi o UNO, com processador Atmega 328, o qual possui 32KB de memória flash, dentre os quais 0,5KB são utilizados pelo *bootloader*, sendo que este permite enviar novos programas à placa sem o uso de um programador de *hardware* externo, o que limita o desenvolvimento da programação do arduino.

Visando a aplicação futura do projeto em todas as salas de aula, foi necessária a definição de uma *label* para um componente constante de identificação da localização do módulo leitor (componente “SALA”, ilustrado na linha 1 do quadro 3), para que o sistema desenvolvido possa realizar o filtro e identificação da posição de cada professor. O projeto utiliza algumas variáveis para desenvolvimento do *firmware*, como a variável do tipo char “leituraTAG”, a qual contém o comando para leitura do serial contido na TAG e “searchRES”, utilizada para armazenar o resultado enviado pelo módulo como resposta a solicitação enviada, disposto nas linhas 5 e 7 do quadro 3, respectivamente.

Quadro 3 - Definição de componentes e variáveis.

```
1. #define SALA  502
2. #define ledPin 7
3. int val = 0,i=0;
4. int status = 0;
5. unsigned char leituraTAG[] = {
6. 0xAA, 0xBB,0x02, 0x20, 0x22};
7. unsigned char searchRES[4];
```

Fonte: Do autor.

O tratamento da leitura das TAGs é de responsabilidade do *firmware*. Assim, para realizar a leitura do código contido nas TAGs, foi necessária a criação de uma função que irá verifica se alguma TAG foi encontrada (função "searchCard"), ilustrado na linha 1 do quadro 4; quando encontrada alguma TAG é enviado o comando "rfid.write(leituraTAG, 5)", ilustrado na linha 2 do quadro 4, o qual escreve os dados da variável "leituraTAG" e envia ao microcontrolador via comunicação serial. Quando o módulo RFId retorna alguma resposta, esta é armazenada em uma variável descrita no comando "val = rfid.read()", linha 7 do quadro 4. Após, o programa executa um laço para percorrer os valores obtidos como resposta e verifica se esta representa uma falha ou se obteve sucesso no retorno e, por fim, armazena o serial da TAG para ser enviado ao monitor serial.

O código fonte, completo, do *firmware* encontra-se ilustrado no apêndice A.

Quadro 4 - Comando para leitura de TAGs.

```
1.      int searchCard(){
2.       rfid.write(searchCMD, 5);
3.       delay(100);
4.       status = 0;
5.       while(true) {
6.         if (rfid.available() > 0) {
7.            val = rfid.read();
8.            switch (status) {
9.               […]
10.             }
11.         }
12.       }
13.     }
```

Fonte: Do autor.

Recursos

Os recursos utilizados para este projeto estão relacionados à aquisição dos materiais, conforme tabela 5.

Tabela 5 - Descrição dos recursos para implementação do estudo de caso.

	Recursos		
Material	**Quantidade**	**Valor Unitário**	**Valor Total**
Placa Arduino UNO	1	R$ 68,00	R$ 68,00
Módulo RFID YHY502CTG	1	R$ 181,50	R$ 181,50
Cartão RFId 13.56MHz MIFARE 1Kb	1	R$ 4,50	R$ 4,50
Chaveiro RFId 13.56MHz MIFARE 1Kb	1	R$ 6,50	R$ 6,50
Bateria 9V	1	R$ 9,90	R$ 9,90
Total			R$ 270,40

Fonte: Do Autor.

3.3.2 Camada Intermediária

Esta sessão descreve a camada responsável pela interação entre as camadas de usuário e física, onde foi estabelecida a comunicação, bem como o desenvolvimento do *middleware* e a modelagem do banco de dados.

Middleware

O *middleware* deste projeto tem como função fazer a comunicação entre o servidor (local onde todos os serviços estarão rodando) e a estrutura física. Esta camada foi dividida em duas partes: a primeira, composta pelo *firmware* (descrito na sessão anterior), é responsável pelo controle da estrutura física, e a segunda parte é responsável pela identificação dos leitores e recuperação dos dados enviados pelos microcontroladores. Porém para o desenvolvimento deste projeto, foi utilizado apenas um microcontrolador (descrito na sessão 3.3.1), tornando os dados manipuláveis pelo sistema, conforme detalhado a seguir.

O *middleware* desenvolvido segue a seguinte estrutura: a leitura da porta serial do computador é realizada em modo *daemon*, ou seja, o programa está sempre em execução, quando algum dado é enviado do microcontrolador pela porta serial, o *middleware* recebe os dados e trata-os, armazenando no banco de dados (descrito nesta sessão, item banco de dados).

O desenvolvimento deste *middleware* foi realizado com o uso da IDE Netbeans, na linguagem de programação Java, além da conexão com o banco de dados PostgreSQL. O código desta aplicação foi baseado em um monitor serial distribuído, sob licença *open-source*, pela própria Arduino.

- Biblioteca de comunicação serial

O microcontrolador utiliza comunicação serial para envido dos dados, portanto, o sistema foi elaborado estabelecendo a leitura dos dados recebidos pela porta serial. Para isso, foi necessário utilizar da biblioteca RXTX que, através de suas funções, permite comunicação serial do computador com o arduino, na linguagem de programação Java.

Para utilização da biblioteca é necessário informar no código, a porta COM que o microcontrolador está utilizando (ilustrado na linha 4 do quadro 5) e a taxa de transmissão que a porta COM deve utilizar, em bits/segundo (linha 9, quadro 5).

Quadro 5 - Uso da biblioteca RXTX.

```
1. private static final String PORT_NAMES[] = {
2.     "/dev/tty.usbserial-A9007UX1",
3.     "/dev/ttyUSB0",
4.      "COM4",
5. };
6. private BufferedReader input;
7. private OutputStream output;
8. private static final int TIME_OUT = 2000;
9. private static final int DATA_RATE = 9600;
```

Fonte: Do autor.

- Desenvolvimento do *Middleware*

Basicamente o código é dividido em três classes Java, conforme segue:

- Classe CriaConexao.java

Responsável pela conexão com o banco de dados PostgreSQL. A conexão foi estabelecida com o banco de dados de nome “localizacao_rfid”, conforme disposto no apêndice B.

- Classe LocalizacaoDAO.java

Esta classe é responsável pelas funções e persistência do objeto no banco de dados, onde será realizada a inserção de dados e as buscas necessárias no banco de dados. Tal classe encontra-se disposta no apêndice C.

- Classe SerialComm.java

Esta classe é a responsável pela comunicação entre as camadas física e intermediária, pois realiza comunicação com a interface serial, extração dos dados recebidos e inserção dos dados no banco de dados, através da utilização das funções criadas na classe LocalizacaoDAO.java.

A classe SerialComm.java (disponível no apêndice D) com seu método “SerialEvent”, realiza a leitura dos dados e estabelece a conexão com o banco de dados. Conforme especificado na sessão 3.3.1 (item *firmware*), os dados recebidos na porta serial possuem a seguinte estrutura [SERIAL_TAG_PROFESSOR;NUMERO_SALA]. Logo, quando o daemon recebe algum evento de leitura do arduino, o buffer de entrada da comunicação serial é armazenado em uma String onde é realizada então uma “divisão” desta string pelo caractere “;”. Esta divisão é feita com o método split do objeto String, ilustrado na linha 8 do quadro 6, obtendo assim, o id da TAG e o número da sala em variáveis separadas, facilitando a manipulação.

Com estes dados, realiza-se uma busca para verificar se os dados estão cadastrados no banco de dados, ou seja, se a TAG é válida neste sistema; verificando também se a sala recebida está no banco de dados (linha 16, quadro 6) e se a TAG recebida está cadastrada para algum professor (linha 11, quadro 6). Se esta verificação é válida, os dados são então inseridos no banco de dados, na tabela de nome “localizacao” (linha 26, quadro 6), indicando o professor, sala, data e hora que a localização foi registrada, conforme mostra o quadro 6.

Quadro 6 - Função SerialEvent.

```
1.    […]
2.    public synchronized void serialEvent(SerialPortEvent oEvent) {
3.    if (oEvent.getEventType() == SerialPortEvent.DATA_AVAILABLE) {
4.      try {
5.          readSerial = input.readLine();
6.          CriaConexao criaConexao = new CriaConexao();
7.          LocalizacaoDAO localizacao = new
LocalizacaoDAO(criaConexao.getConexao());
8.          String serial[] = readSerial.split(";");
9.          setLeituraTAG(serial[0]);
10.         readSala = Integer.parseInt(serial[1]);
11.         int loc_professor = localizacao.buscarTag(leituraTAG);
12.         setProfessor(0);
13.         if (loc_professor != 0){
14.            setProfessor(loc_professor);
15.         }
16.         int loc_sala = localizacao.buscarSala(readSala);
17.         setSala(0);
18.         if (loc_sala != 0){
19.           setSala(loc_sala);
20.         }
21.     dataAtual = Calendar.getInstance().getTime();
22.     Date hora = Calendar.getInstance().getTime();
23.     SimpleDateFormat formatoHora = new SimpleDateFormat("HH:mm:ss");
24.     setDataAtual(dataAtual);
25.     setHoraAtual(formatoHora.format(hora));
26.     localizacao.inserir(this);
27.    }
28.  }
```

Fonte: Do autor.

Banco de Dados

O banco de dados utilizado no projeto foi planejado para cadastro e armazenamento dos dados. Esta camada possui, também, função de interfaceamento entre a camada intermediária e de usuário, pois o *middleware* envia os dados registrados no banco de dados e a aplicação final busca os dados no mesmo, ilustrado na figura 12. Para o estudo de caso proposto, foi utilizado o banco de dados PostgreSQL.

Figura 12 - Banco de dados, interface entre camadas.

Fonte: Do autor.

- Requisitos do banco de dados

Para definição dos requisitos necessários é importante conhecer as funcionalidades atuais e o que se pretende aperfeiçoar. Para isto, buscou-se compreender como são feitas as buscas de professores atualmente. Por exemplo:

- Qual a localização do professor A? ou;
- Quais professores do curso TSPI estão no câmpus?

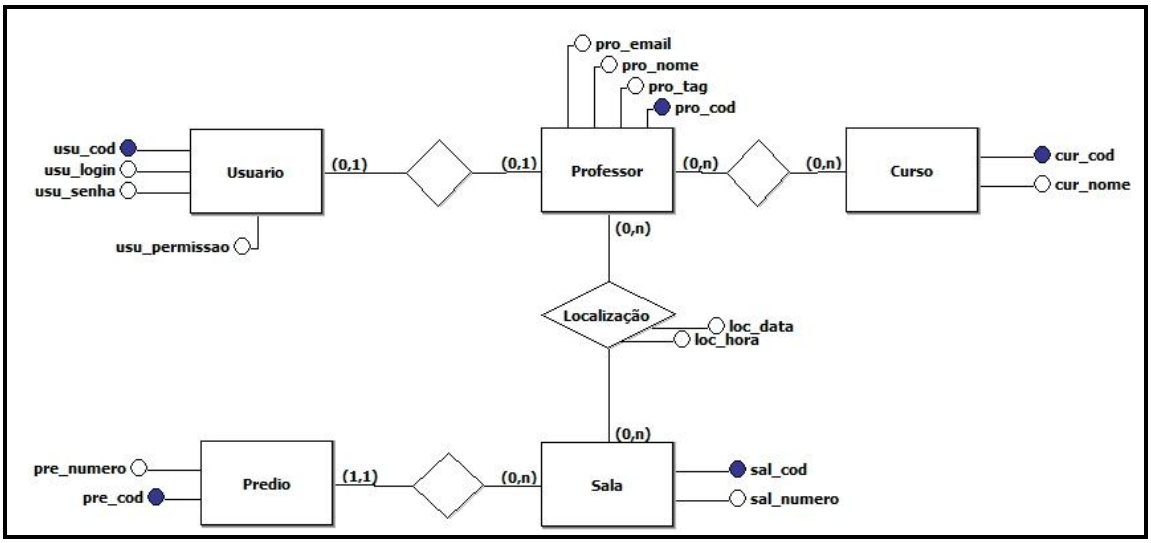
Para isso foi realizado um levantamento do funcionamento atual, realizando uma modelagem visando à localização dos professores e também a realização das buscas, assim verificou-se que cada curso possui vários professores e cada professor pode ministrar aula em vários cursos, assim como, as aulas poderão ser ministradas em diversas salas de aula, e várias salas de aula compõem um prédio. Logo, um professor pode estar localizado em uma determinada sala de aula em um determinado horário e, no momento seguinte, estar em uma sala diferente.

O sistema deverá possuir controle de acesso, controlado por usuário, onde este usuário poderá ou não estar vinculado a algum professor.

- Modelagem do Banco de Dados

Para a construção do banco de dados foi utilizado a ferramenta BrModelo[9], com a qual foi realizada a modelagem ER, buscando atender os requisitos citados no item anterior, ilustrada na figura 13.

Figura 13 - Modelagem ER do Sistema.



Fonte: Do autor.

A modelagem é detalhada a seguir:

- **prédio:** Cada prédio é composto por um número e um código, que representa sua identificação. Cada prédio pode conter várias salas de aula.
- **sala:** A sala é composta pelo número e um código identificador. Como a relação entre sala e prédio é 1-N (um para muitos), a tabela sala possuirá uma chave estrangeira de prédio.
- **usuário:** Esta tabela possui somente os dados para acesso ao sistema, sendo eles: login, senha, permissão e um código identificador.
- **professor:** A tabela "professor" é composta pelo nome e e-mail. Assim, como o sistema é baseado na tecnologia RFId, onde cada professor possuirá uma TAG única, a tabela "professor" deverá conter o registro do serial da TAG. Além disso, a relação entre a tabela "professor" e "usuario" é 1-1 (um para um), a qual propõe que o professor poderá conter um usuário, assim como o usuário não precisará ser de um

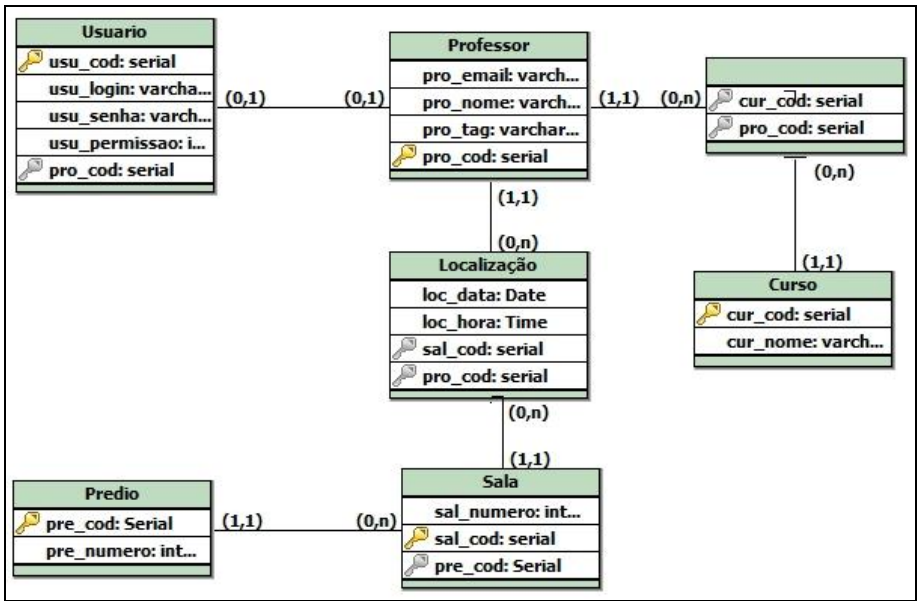
[9] http://sis4.com/brModelo/

professor. Logo, a tabela “professor” possui uma chave estrangeira, não obrigatória, para a tabela “usuario”. Já a relação das tabelas “professor” e “sala”, N-N (muito para muitos), mostra que a sala de aula poderá possuir mais de um professor, assim como o professor pode ministrar aula em várias salas. Logo, para esta relação gerou-se outra tabela: “localizacao”.

- **localização:** A tabela “localizacao” apresenta o registro de uma localização em determinado momento por um professor, a qual é composta por: sala, professor, data e hora do registro. Esta tabela será alimentada pelo *middleware* da aplicação, descrito anteriormente, nesta sessão.
- **curso:** Composta pelo nome e código identificador do curso. Este possui relacionamento com a tabela “professor”, sendo esta N-N (muito para muitos), assim verifica-se que o professor pode ministrar aula em vários cursos, assim como, o curso poderá possuir vários professores em sua composição. Tal relação gerou a tabela “professor_por_curso”.
- **professor_por_curso:** Composta pelo curso e professor, permitindo assim a relação estabelecida.

Tal modelagem, realizada com o modelo conceitual, encontra-se ilustrada na figura 14.

Figura 14 - Modelagem Conceitual do Sistema.



Fonte: Do autor.

3.3.3 Camada Usuário

Trata-se da camada de mais alto nível, onde está a aplicação final desenvolvida para o projeto, e que, oculta o funcionamento das camadas inferiores, ao mesmo tempo em que permite ao usuário fazer uso do sistema localizando as pessoas.

- Aplicação

A aplicação é resultante de consultas SQL, mostrando as últimas localizações dos professores. Sua estrutura é composta pela seguinte lógica: o usuário seleciona a busca desejada, assim os dados são recuperados da camada intermediária (banco de dados), tornando as informações disponíveis ao usuário final.

Seu desenvolvimento foi realizado na linguagem de programação PHP 5.5.3, com Apache Server 2.4.4, instalados através do programa XAMPP e o desenvolvimento foi realizado através da IDE Netbeans, já a conexão foi estabelecida com o banco de dados PostgreSQL. Para a realização desta aplicação, foi utilizado também à linguagem de marcação HTML e estilos CSS para o design.

O desenvolvimento da aplicação foi elaborado em arquivos e páginas principais, dispostas através de páginas Web acessíveis através de um menu, onde encontra-se a página Pesquisa Professor, a qual corresponde também a página principal da aplicação, página Pesquisa por Curso e a página Pesquisa por Prédio, as quais encontram-se descritas a seguir:

- index.php

O arquivo índex é responsável pela conexão com o banco de dados e a estruturação das páginas. Através deste arquivo se constituiu a estrutura utilizada nas demais páginas da aplicação, nela se estabeleceu o design e estrutura da sua composição, como: topo, links de navegação e rodapé. Logo, somente o conteúdo das demais páginas é alterado. O código completo desta página encontra-se apresentado no apêndice E.

A conexão com o banco de dados é responsável pela recuperação dos dados enviados pelo *Middleware* ao Banco de Dados (itens já discutidos nas sessões anteriores), logo a interação entre o banco de dados e o PHP foi desenvolvido conforme linha 6 do quadro 7.

Quadro 7 - Estabelecimento da conexão com o banco de dados.

```
1. $servidor = "localhost";
2. $porta = 5432;
3. $bancoDeDados = "localizacao_rfid";
4. $usuario = "postgres";
5. $senha = "123456";
6. $conexao = pg_connect("host=$servidor
   port=$porta dbname=$bancoDeDados
   user=$usuario password=$senha");
```

Fonte: Do Autor.

- Página Pesquisa Professor:

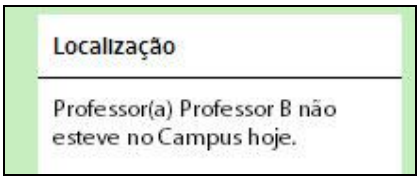
Figura 15 - Layout da aplicação: página Pesquisa Professor.

Fonte: Do autor.

Conforme ilustrado na figura 15, o usuário seleciona o professor que deseja localizar e o sistema mostra os últimos quatro registros de localizações na data atual. Portanto, esta

página realiza buscas através de consultas SQL, assim como trata as informações caso o resultado da consulta não retorne registro de localizações, conforme ilustrado na figura 16.

Figura 16 - Tratamento de saída de informação: página Pesquisa Professor.



Fonte: Do Autor.

O arquivo inc_pesquisa.php está atribuído à página Pesquisa Professor. Para o desenvolvimento deste, utiliza-se o arquivo funcoes.php (descrito no apêndice F), responsável pelas funções utilizadas na aplicação, as quais possuem consultas SQL do banco de dados. O código fonte, completo, desta página encontra-se no apêndice G.

- Página Pesquisa por Prédio

O arquivo inc_pesquisa_predio.php está atribuído a página Pesquisa por Prédio, onde o usuário seleciona o prédio que deseja monitorar, logo é realizada a busca dos professores presentes no prédio selecionado, de acordo com a última localização registrada por cada um, e mostra na tela a última localização de cada professor, quando esta é referente ao presente prédio (o layout da página se encontra ilustrado na figura 17). O código fonte, completo, desta página encontra-se descrita no apêndice H.

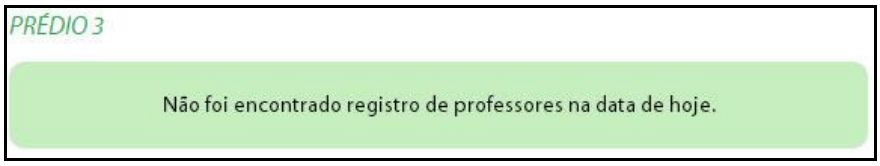
Figura 17 - Layout da aplicação: página Pesquisa por Prédio.

Fonte: Do autor.

As buscas são realizadas através de consulta SQL estabelecidas com o banco de dados, assim como a página Pesquisa Professor, esta página também utiliza funções incluídas pelo arquivo funcoes.php.

Caso a execução da consulta SQL não retorne registros, o sistema mostrará uma mensagem ao usuário, a fim de facilitar o entendimento (ilustrado na figura 18).

Figura 18 - Tratamento de saída de informação: página Pesquisa por Prédio.



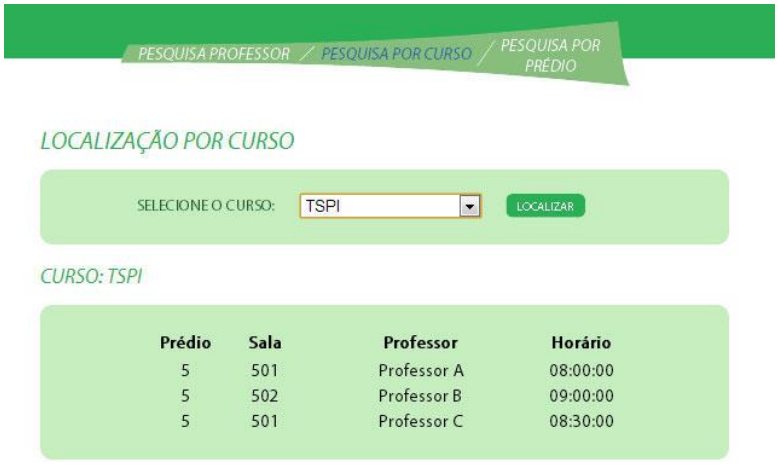
Fonte: Do autor.

- Página Pesquisa por Curso:

A página de Pesquisa por Curso (arquivo inc_pesquisa_curso.php e layout ilustrado na figura 19) segue o mesmo layout da página Pesquisa por Prédio alterando somente as consultas SQL realizadas, conforme apêndice I. Assim, na página Pesquisa por Curso, o usuário seleciona o curso que deseja pesquisar e uma lista com os professores que possuem vínculo com este, contendo sua localização (prédio, sala, nome do professor e horário do último registro de localização), é mostrada na tela. Caso nenhum professor vinculado ao curso possua localização na data atual, uma mensagem é mostrada com a informação.

No desenvolvimento deste código são utilizadas funções PHP, através do arquivo funcoes.php.

Figura 19 - Layout da aplicação: página Pesquisa por Curso.

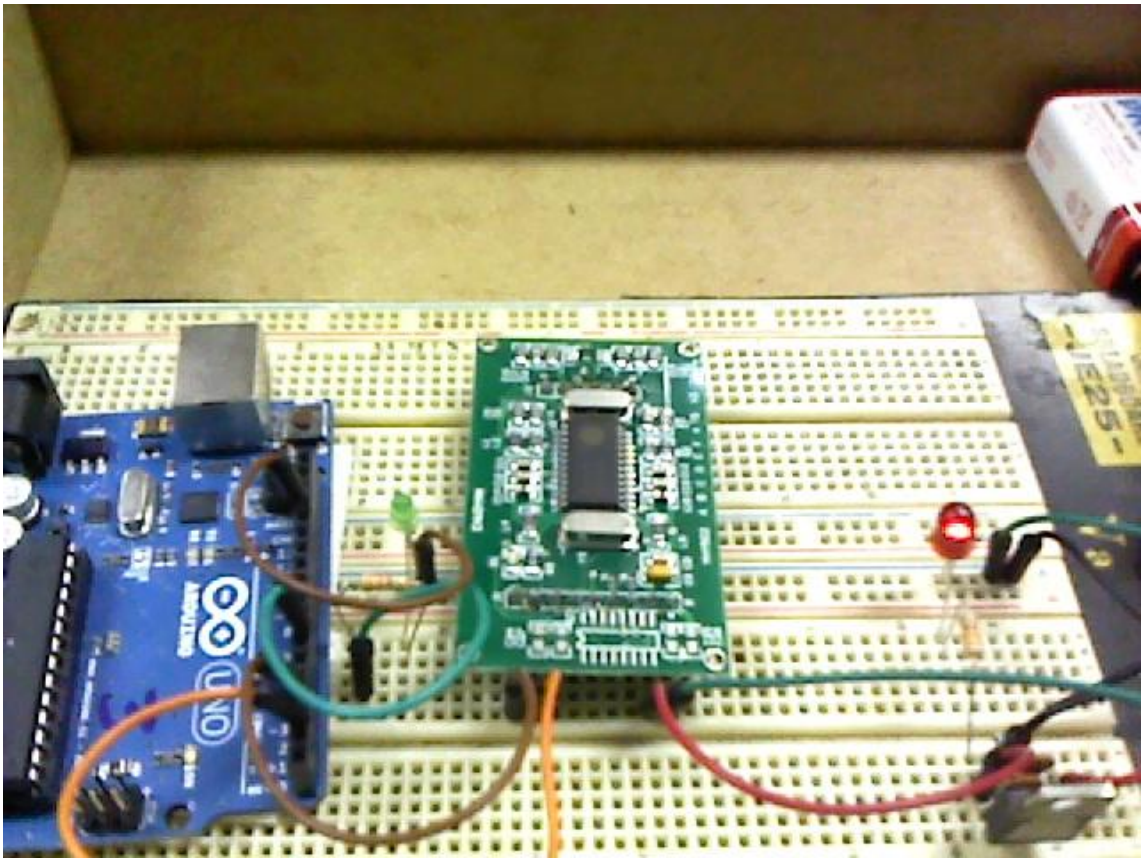


Fonte: Do Autor.

3.4 AMBIENTE DE TESTE

Após o desenvolvimento do protótipo foram realizados os devidos testes, onde o cenário do mesmo consistiu-se em um circuito da estrutura física ligada em uma protoboard, conforme ilustrado na figura 20.

Figura 20 - Ambiente de prototipação.



Fonte: Do autor.

De acordo com o *firmware* desenvolvido (sessão 3.3.1), a cada leitura da TAG no módulo RFId, a led verde acende, constatando visualmente está leitura. Foram realizados testes baseados nas duas TAGs adquiridas, e após várias identificações de cada uma das TAGs no módulo leitor RFId, verificou-se que a TAG em formato de Cartão RFId possui um alcance máximo de 5 cm, pouco maior que a TAG em formato de chaveiro, a qual possui um alcance máximo de 4 cm.

Outra constatação com os testes realizados foi a influência da angulação da face da TAG em relação à face do módulo leitor, a qual dependendo da angulação em que se encontra altera o campo de leitura criado.

Os testes foram executados em somente um ambiente de prototipação onde o número da sala corresponde a sala 502, do prédio 5, e os professores correspondem a dois professores testes. Através deste ambiente de teste foi possível verificar a precisão do sistema, onde utilizando a tecnologia RFId foi identificado uma precisão grande, pois não houve erros na aplicação, porém o sistema depende da identificação do usuário.

CONSIDERAÇÕES FINAIS

Com a implementação do estudo de caso obteve-se como resultado a efetiva geolocalização de pessoas em ambientes fechados; no caso, nos prédios do câmpus do instituto. Ainda, devido a forma com que a TAG RFId é detectada, o sistema mostrou possuir alta precisão; e ainda, nos testes executados, nenhuma falha de leitura e localização foram detectadas.

O projeto desenvolvido atendeu plenamente o objetivo de baixo custo, onde, devido aos componentes físicos utilizados, resultou em um sistema economicamente acessível. Além disso, destaca-se a efetiva integração de diversas tecnologias e linguagens de programação, que se revelaram eficientes no funcionamento do sistema, uma vez que foi esta interação que tornou possível esta implementação.

Outra tática abordada, que gerou resultados positivos, foi o armazenamento dos dados referentes aos registros de localização, já que estes poderão ser utilizados posteriormente para várias aplicações – apresentadas adiante.

Além disso, o sistema se mostrou relativamente simples, onde todos os processos são intuitivos, desde a detecção da geolocalização de pessoas, onde a pessoa simplesmente aproxima a TAG do leitor, até a visualização dos dados, pois possui uma interface amigável, permitindo que usuários leigos consigam fazer buscas com extrema facilidade.

Porém, ao longo do desenvolvimento encontraram-se algumas dificuldades, as quais estão descritas a seguir.

- Alcance de leitura da tecnologia: houve um franco desacordo entre o alcance especificado nas referencias bibliográficas com o alcance realmente obtido na implementação. O alcance real dos módulos RFId, do tipo HF, foram muito abaixo da proposta inicial do projeto, sendo necessário fazer adaptações no sistema.
- Campo de leitura: o ângulo em que a TAG é apresentada ao leitor RFId influencia no campo de leitura, podendo este chegar a ser nulo, impedindo a leitura.
- Alcance de leitura X Custo da implementação: O alcance do módulo utilizado é pequeno; para ampliar este alcance seria necessário o uso de leitores de alto desempenho, que implicaria em um aumento significativo de custo. Além disto, devido ao problema causado pela angulação no campo de leitura, seria necessária a

aquisição de três antenas por porta de sala, distribuídas entre as laterais e a parte superior de cada porta, aumentando novamente os custos.

Para resolver estes problemas encontrados na técnica de identificação por aproximação (de forma automática), optou-se por adoção do método "observação de sistemas de identificação automática", também da técnica de proximidade, onde a pessoa declara sua localização, evitando assim os problemas citados acima.

Por fim, o aprimoramento deste projeto é possível através da implementação de algumas memórias em trabalhos futuros, tais como:

- CRUD: seria interessante o desenvolvimento de um CRUD para inclusão das informações, tais como: cadastro de professores, usuários, prédio e salas de aula.
- Abertura de portas mediante identificação: considerando que o sistema foi desenvolvido exigindo identificação por aproximação, e que o professor deverá se identificar ao chegar na sala, pode-se também incluir um mecanismo de abertura automática de portas mediante identificação.
- Controle do tamanho das tabelas: Como a tabela de localização será constantemente alimentada, esta se tornará gigantesca, sendo necessário uma *trigger* para automatizar o processo de *backup* e exclusão de dados antigos.
- Leitores RFId com comunicação por rede: uso de *shields* de comunicação de rede ao invés de usar comunicação serial para atualização dos dados no servidor de aplicação.
- Controle de saída: realizar o controle de entrada e saída de professores nas salas de aula.

REFERÊNCIAS

ARDUINO. Arduino. Disponível em: <http://arduino.cc/>. Acesso em: 20 mai. 2013.

Arduino UNO. Datasheet Atmega 328P. 2009. Disponível em: <http://www.atmel.com/Images/doc8161.pdf>. Acesso em 29 ago. 2013.

ATMEL Corporation. *Understanding the Requirements of ISO/IEC 14443 for Type B Proximity Contactless Identification Cards* (2005). Disponível em: <http://www.atmel.com/images/doc2056.pdf/>. Acesso em: 20 out. 2013.

Biblioteca SerialSoftware.h. Disponível em: < http://pt.scribd.com/doc/92033530/do-a-Programar-Em-Arduino>. Acesso em 20 set. 2013.

BORGES, Guilherme Antonio. *Arquitetura ubíqua para ambientes residenciais*. 2012. 79f. Monografia (Graduação em Tecnologia em Sistemas para Internet). Instituto Federal Sul-Rio-Grandense, Câmpus Passo Fundo, Passo Fundo, 2012. Disponível em: <http://inf.passofundo.ifsul.edu.br/graduacao/monografias-defendidas/2012-1/GuilhermeBorges.pdf>. Acesso em: 20 mai. 2013.

BRASIL. Resolução nº 506, de 1º de julho de 2008. Disponível em: <http://legislacao.anatel.gov.br/resolucoes/23-2008/104-resolucao-506>. Acesso em 17 abr. 2013.

DIAS, Renata Rampim de Freitas; BALADEI, Suely De Pieri. Diferenças entre as frequências do sistema RFId passivo. *RFID Journal Brasil*. jun 2012. Disponível em: <http://brasil.rfidjournal.com/artigos/vision?9591>. Acesso em: 15 abr. 2013.

DIAS, Renata Rampim de Freitas. A importância de um middleware para o sistema RFId. *RFID Journal Brasil.* set. 2012. Disponível em: <http://brasil.rfidjournal.com/artigos/vision?9940>. Acesso em: 17 abr. 2013.

GARCIA, Cristiano P. *Sistemas de Localização Indoor e Outdoor*. 2009. 14f. Monografia (Graduação em Ciências da Computação). Universidade de São Paulo, 2009. Disponível em: <http://grenoble.ime.usp.br/~gold/cursos/2008/movel/grad/monografiaCristianoGarcia.pdf>. Acesso em 10 mai. 2013.

GINES, Fernando Henrique; TSAI, Thiago Tadeu. *Projeto e Implementação de um sistema de identificação por RFId para uma aplicação de automação residencial*. 2007. 79f. Monografia (Graduação em Engenharia da Computação) - Escola Politécnica da Universidade de São Paulo, São Paulo, 2007. Disponível em <http://www.pcs.usp.br/~pcspf/2007/Cooperativo%202007/PCS%202050%20COOP%20Grupo%20(18)/grupo18c.pdf>. Acesso em: 04 abr. 2013.

HIGHTOWER, Jeffrey; BORRIELLO, Gaetano. A Survey and Taxonomy of Location Systems for Ubiquitous Computing. 2001. Disponível em:

<http://www.csd.uoc.gr/~hy439/lectures11/hightower2001survey.pdf>. Acesso em 25 jun. 2013.

LIMA, Eliomar Araújo de. Sistemas para localização de pessoas e objetos em ambientes *indoor*. 2001. Disponível em: <http://www-di.inf.puc-rio.br/~endler/courses/Mobile/Monografias/01/Eliomar-Caching.doc>. Acesso em 10 mai. 2013.

LOUREIRO, Antonio A. F et al. Rede de sensores Sem Fio. 2003. Disponível em: <http://homepages.dcc.ufmg.br/~loureiro/cm/docs/sbrc03.pdf>. Acesso em 26 abr. 2013.

MCROBERTS, Michael. Arduino Básico. [tradução Rafael Zanolli]. São Paulo: Novatec Editora, 2011.

MENDES, Gláucia Maria Pastro. *Sistema mobile web para busca georreferenciada de imóveis*. 2011. 42f. Monografia (Especialização em Tecnologia Java) - Universidade Tecnológica Federal do Paraná, Curitiba, 2011. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/622/1/CT_JAVA_VI_2010_08.PDF>. Acesso em 09 mai. 2013.

MONK, Simon. 30 Arduino™ Projects for the Evil Genius™. United States: McGraw-Hill, 2010.

MOTA, Rafael Perazzo Barbosa. *RFId - Radio Frequency Identification*. 2012. 21f. Monografia (Pós-Graduação em Ciências da Computação) - Universidade de São Paulo, São Paulo, 2012. Disponível em <http://grenoble.ime.usp.br/~gold/cursos/2012/movel/mono-1st/2505-1_RafaelPerazzo.pdf>. Acesso em: 12 abr. 2013.

National Instruments. Parte 2: Surge o novo padrão de RFId EPC Gen2. 2011. Disponível em: < http://www.ni.com/white-paper/13220/pt >. Acesso em 03 jun. 2013.

OLIVEIRA, Pedro Nuno Fontes de; TAVARES, Ricardo Meireles. Análise da Linha de Produção e Estudo da Automação de Processos-PE10. Disponível em: <http://paginas.fe.up.pt/~ee95203/>. Acesso em: 20 ago. 2013.

PASSARETTI, Caio Santi. *RFID – Identificação por radiofrequência movendo-se para o futuro*. 2008. 121f. Monografia (Graduação em Engenharia Elétrica) - Universidade de Brasilia, Brasilia, 2008. Disponível em: <http://bdm.bce.unb.br/bitstream/10483/901/1/2008_CaioSantiPassaretti.pdf>. Acesso em 20 ago. 2013.

PRADO, Caio Vinicius Domingos do. *Framework Web para Automação.* 2012. 51f. Monografia (Graduação em Ciência da Computação) - Universidade Vila Velha, Vila Velha, 2012. Disponível em: <http://www.uvv.br/edital_doc/Framework%20Web%20para%20Automa%C3%A7%C3%A3o.pdf>. Acesso em 21 mai. 2013.

PRATA, Pedro Isidoro. *Sistemas de Localização para Ambientes Interiores baseados em RFId*. 2008. 150f. Monografia (Mestrado em Engenharia

Eletrônica e Telecomunicações) - Universidade de Aveiro, Aveiro, 2008. Disponível em: <http://ria.ua.pt/bitstream/10773/1906/1/2008001651.pdf>. Acesso em: 15 abr. 2013.

RABELLO, Klaus Denecke. O futuro do código de barras (RFId). 2012. Disponível em: <http://4b-2012-02.bligoo.com.br/o-futuro-do-codigo-de-barras-rfid>. Acesso em: 10 mai. 2013.

RFID Journal Brasil. Perguntas Frequentes. Disponível em: <http://brasil.rfidjournal.com/perguntas-frequentes>. Acesso em: 16 mai. 2013.

RFID WORLD. Understanding Passive RFId (Radio Frequency Identification) Technology. Disponível em:<http://www.rfidworld.ca/understanding-passive-rfid-radio-frequency-identification-technology/1294>. Acesso em 14 jun. 2013.

SOUZA, Carlos Danilo Rosa de; SILVA, Marcelo Wanderley Santos da; BARROSO, Paulo Henrique Carvalho. *RFID – Identificação Por Radiofrequência*. 2010. 76f. Monografia (Graduação em Ciência da Computação). Universidade da Amazônia, Belém, 2010. Disponível em: <http://www.bcc.unama.br/index.php?option=com_docman&task=doc_details&gid=64&Itemid=76 >. Acesso em: 23 out. 2013.

TANENBAUM, Andrew S; WETHERALL, David. Redes de Computadores. Tradução Daniel Vieira. Ed. 5. São Paulo: Pearson Prentice Hall, 2011.

ZANLOURENSI , Luis Guilherme. Identificação por Rádio Freqüência - RFId. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/661/1/CT_TELEINFO_XIX_2011_14.pdf>. Acesso em: 16 mai. 2013.

YHY502CTG Datasheet. YHY502CTG++ 13.56MHz RFID Mifare® Read/Write Module. 2010. Disponível em: <http://www.ehuoyan.com/download/module//DS_YHY502CTGV33.pdf>. Acesso em 29 ago. 2013.

APÊNDICES

APÊNDICE A – Código fonte do *Firmware.*

```
#include <SoftwareSerial.h>
#define SALA  502
#define ledPin 7 //pino para acender o led
int val = 0,i=0;
int status = 0;
unsigned char searchCMD[] = { //dados enviados pela porta serial (pinos sw serial)
  0xAA, 0xBB,0x02, 0x20, 0x22};
unsigned char searchRES[4];
SoftwareSerial rfid(2, 3); //portas seriais
void setup()
{
  Serial.begin(9600);
  rfid.begin(19200); //iniciando a serial com velocidade 19200
  pinMode(ledPin, OUTPUT);
}
int searchCard() //procurar TAG
{
  rfid.write(searchCMD, 5);  //escrevendo dados para transmitir pela porta serial
  delay(100);
  status = 0;
  while(true) {
    if (rfid.available() > 0) {
      val = rfid.read(); //retorna o dado recebido pelo pino rx
     switch (status) { //pega o dado do leitor
      case 0:
        if (val == 0xAA) status = 1;
        break;
      case 1:
        if (val == 0xBB) status = 2;
        else return -1; //erro
        break;
      case 2:
        if (val == 0x06) status = 3;
        else return -1;
        break;
      case 3:
        if (val == 0x20) status = 4;
        else return -1;
        break;
      case 4:
      case 5:
      case 6:
      case 7:
        searchRES[status - 4] = val; //passando valor lido para ser mostrado depois
        status ++;
        break;
      case 8:
        return 0; // sucesso
        break;
      default:
        return -1;
        break;
      }
    }
  }
}
```

```
void loop()
{
  if (searchCard() != -1) {
    digitalWrite(ledPin, HIGH);
    for(i=0; i<4; i++){
      Serial.print(searchRES[i],HEX);
      Serial.print("|");
    }
    Serial.print(";");
    Serial.print(SALA,DEC);
    Serial.print("\n");
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(1000);
  }
}
```

APÊNDICE B – Arquivo CriaConexao.java.

```
package Util;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class CriaConexao {
    public CriaConexao() {
        try {
            Class.forName("org.postgresql.Driver");
        } catch (ClassNotFoundException ex) {
            System.out.println("Erro ao carregar o driver do Postgres." + ex);
        }
    }
    public Connection getConexao() {
        Connection conexao = null;
        try {
            conexao = DriverManager.getConnection("jdbc:postgresql://localhost/localizacao_rfid", "postgres",
"123456");
        } catch (SQLException ex) {
            System.out.println("Erro ao executar conexão com o banco de dados." + ex);
        }
        return conexao;
    }
}
```

APÊNDICE C – Arquivo LocalizacaoDAO.java.

```
package dao;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Time;
import modelo.SerialComm;
public class LocalizacaoDAO {
    private Connection conn;
    private PreparedStatement pstmt;
    private ResultSet rs;
    public LocalizacaoDAO(Connection conexao) {
        this.conn = conexao;
    }
    public void inserir(SerialComm serial) {
        String sql = "INSERT into localizacao (sala, professor, loc_data, loc_hora) values (?, ?, ?, ?)";
        try {
            pstmt = conn.prepareStatement(sql);
            pstmt.setInt(1, serial.getSala());
            pstmt.setInt(2, serial.getProfessor());
            pstmt.setDate(3, new java.sql.Date(serial.getDataAtual().getTime()));
            pstmt.setTime(4, Time.valueOf(serial.getHoraAtual()));
            pstmt.execute();
            pstmt.close();
        } catch (SQLException e) { System.out.println("Erro ao inserir Leitura da Tag: " + e);  }
    }
    public int buscarTag(String readTAG) {
        int professor = 0;
        String sql = "SELECT * FROM professor p WHERE pro_tag = ?";
        try {
            pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, readTAG);
            rs = pstmt.executeQuery();
            while (rs.next()) {
                professor = rs.getInt("pro_cod");
            }
            pstmt.close();
        } catch (SQLException e) { System.out.println("Erro ao buscar TAGs dos professores: " + e); }
        return professor;
    }
    public int buscarSala(int readSala) {
        int sala = 0;
        String sql = "SELECT * FROM sala WHERE sal_numero = ?";
        try {
            pstmt = conn.prepareStatement(sql);
            pstmt.setInt(1, readSala);
            rs = pstmt.executeQuery();
            while (rs.next()) {
                sala = rs.getInt("sal_cod");
            }
            pstmt.close();
        } catch (SQLException e) { System.out.println("Erro ao buscar Sala: " + e); }
        return sala;
    }
}
```

APÊNDICE D – Arquivo SerialComm.java: Comunicação Serial.

```
public class SerialComm implements SerialPortEventListener {
   private String leituraTAG, readSerial;
   private Date dataAtual;
   private String horaAtual;
   private int professor;
   private int sala;
   int readSala;
   SerialPort serialPort;
   /**
    * Portas seriais utilizadas
    */
   private static final String PORT_NAMES[] = {
      "/dev/tty.usbserial-A9007UX1", // Mac OS X
      "/dev/ttyUSB0", // Linux
      "COM4", // Windows
   };
   private BufferedReader input;
   private OutputStream output;
  private static final int TIME_OUT = 2000;
  private static final int DATA_RATE = 9600;
   public void initialize() {
      CommPortIdentifier portId = null;
      Enumeration portEnum = CommPortIdentifier.getPortIdentifiers();
      //Encontra e instancia a porta serial setado em PORT_NAMES.
      while (portEnum.hasMoreElements()) {
         CommPortIdentifier currPortId = (CommPortIdentifier) portEnum.nextElement();
         for (String portaCom : PORT_NAMES) {
            if (currPortId.getName().equals(portaCom)) {
               portId = currPortId;
               break;
          }  }  }
      if (portId == null) {
         System.out.println("Não foi possível localizar a comunicação com a porta Serial");
         return;
      }  try {
         // abre a porta serial e use o nome da classe para o appName.
         serialPort = (SerialPort) portId.open(this.getClass().getName(),
               TIME_OUT);
         // seta os parametros das portas
         serialPort.setSerialPortParams(DATA_RATE,
               SerialPort.DATABITS_8,
               SerialPort.STOPBITS_1,
               SerialPort.PARITY_NONE);
         // open the streams
         input = new BufferedReader(new InputStreamReader(serialPort.getInputStream()));
         output = serialPort.getOutputStream();
         // adiciona evento ao listeners
         serialPort.addEventListener(this);
         serialPort.notifyOnDataAvailable(true);
      } catch (Exception e) { System.err.println(e.toString());  }
   }
public synchronized void close() {
      if (serialPort != null) {
         serialPort.removeEventListener();
         serialPort.close();
      }
   }
```

```
    /**
     * Lidar com um evento na porta serial. Ler os dados e imprimi-los.
     */
    public synchronized void serialEvent(SerialPortEvent oEvent) {
        if (oEvent.getEventType() == SerialPortEvent.DATA_AVAILABLE) {
            try {
                readSerial = input.readLine();
                //Criando conexao
                CriaConexao criaConexao = new CriaConexao();
                LocalizacaoDAO localizacao = new LocalizacaoDAO(criaConexao.getConexao());
                //Separa leitura recebida da Serial entre a TAG e o num da Sala
                String serial[] = readSerial.split(";");
                setLeituraTAG(serial[0]);
                readSala = Integer.parseInt(serial[1]);
                //Localizando professor
                int loc_professor = localizacao.buscarTag(leituraTAG);
                setProfessor(0);
                if (loc_professor != 0){ //Se 0, não foi localizado nenhum professor com o serial da TAG
                    setProfessor(loc_professor);
                }
                //localizando a Sala
                int loc_sala = localizacao.buscarSala(readSala);
                setSala(0);
                if (loc_sala != 0){
                    setSala(loc_sala);
                }
                dataAtual = Calendar.getInstance().getTime();
                Date hora = Calendar.getInstance().getTime();
                SimpleDateFormat formatoHora = new SimpleDateFormat("HH:mm:ss");
                setDataAtual(dataAtual);
                setHoraAtual(formatoHora.format(hora));
                localizacao.inserir(this);
            } catch (Exception e) {
                System.err.println(e.toString());
            }
        }
    }

    public static void main(String[] args) throws Exception {
        SerialComm main = new SerialComm();
        main.initialize();
        Thread t = new Thread() {
            public void run() {
                //o aplicativo aguardará 1segundo,
                //aguardando a ocorrencia de eventos (imprime as mensagens recebidas no console).
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException ie) {
                }
            }
        };
        t.start();
        System.out.println("Executando...");
    }
[...]
```

APÊNDICE E – Arquivo index.php: Estrutura da Aplicação.

```
<?php
require_once './includes/funcoes.php';
$servidor = "localhost";
$porta = 5432;
$bancoDeDados = "localizacao_rfid";
$usuario = "postgres";
$senha = "123456";
//Estabelece a conexão com o banco de dados
$conexao = pg_connect("host=$servidor port=$porta dbname=$bancoDeDados user=$usuario
password=$senha");
if (!$conexao) {
   die("Não foi possível se conectar ao banco de dados.");
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <title> Geolocalização IFSUL </title>
      <link href="css/css_fontes.css" rel="stylesheet" type="text/css" />
      <link href="css/css.css" rel="stylesheet" type="text/css" />
   </head>
   <body>
      <div class="topo"></div>
      <div class="centralizar">
         <div class="menu">
            //Indica que a página principal da aplicação é a página pesquisa (Pesquisa Professor)
            <?php
            @$menu = $_REQUEST['menu'];
            if (empty($menu)) {
               $menu = "pesquisa";
            };
            ?>
            <div class="boxItens">
               <div class="Item1">
               </div>
 <div class="sep1"><img src="imagens/menu/linSeparacao.png" width="13" height="14" /></div>
               <div class="Item2">
                  <a href="pesquisa"><img src="imagens/menu/btnPesquisa.png" name="Pesquisa Por Professor"
width="126" height="12" border="0" id="Pesquisa Por Professor" /></a>
               </div>
      <div class="sep2"><img src="imagens/menu/linSeparacao2.png" width="16" height="22" /></div>
               <div class="Item3">
                  <a href="pesquisa_predio"><img src="imagens/menu/btnPesquisaPredio.png name="Pesquisa por
Prédio" width="83" height="27" border="0" id="Pesquisa por Prédio" /></a>
               </div>
            </div>
         </div>
      </div>
            //incluí o conteúdo da página
            <?php @require_once('includes/abre.php'); ?>
      <div class="rodape">
         <div class="centralizar">
            <p>Trabalho de Conclusão de Curso<br/>
               Aluna: Vanessa Lago Machado</p>
         </div>
      </div>
   </body>
</html>
```

APÊNDICE F – Arquivo funcoes.php: Funções utilizadas na aplicação.

```
<?php
//lista todos os professores para mostrar em uma caixa de seleção (página Pesquisa Professor)
function listaProfessores($conn) {
   $sql = "select pro_cod, pro_nome from professor p
                     order by pro_nome";
   $result = pg_query($conn, $sql);
   $retorno = "<option value=" . "'0'" . ">Selecione...</option>";
   while ($row = pg_fetch_row($result)) {
      $retorno .= '<option value="' . $row[0] . '">' . $row[1] . '</option>';
   }
   return $retorno;
}
//lista todos os prédios para mostrar em uma caixa de seleção (página Pesquisa por Prédio)
function listaPredios($conn) {
   $sql = "select pre_cod, pre_numero from predio
            order by pre_numero";
   $result = pg_query($conn, $sql);
   $retorno = "<option value=" . "'0'" . ">Selecione...</option>";
   while ($row = pg_fetch_row($result)) {
      $retorno .= '<option value="' . $row[0] . '">' . $row[1] . '</option>';
   }
   return $retorno;
}
//lista todos os cursos para mostrar em uma caixa de seleção (página Pesquisa Curso)
function listaCursos($conn) {
   $sql = "select cur_cod, cur_nome from curso order by cur_nome";
   $result = pg_query($conn, $sql);
   $retorno = "<option value=" . "'0'" . ">Selecione...</option>";
   while ($row = pg_fetch_row($result)) {
      $retorno .= '<option value="' . $row[0] . '">' . $row[1] . '</option>';
   }
   return $retorno;
}
//Mostra o nome do professor selecionado na pág. Pesquisa Professor
function mostraNome($conn, $pro_cod) {
   $sql = "select pro_nome from professor
                     where pro_cod = $pro_cod";
   $result = pg_query($conn, $sql);
   while ($row = pg_fetch_row($result)) {
      $nome = $row[0];
   }
   return $nome;
}
```

```
//Mostra o número do prédio selecionado na pág. Pesquisa por Prédio
function mostraNumPredio($conn, $pre_cod) {
   $sql = "select pre_numero from predio
                                                   where pre_cod = $pre_cod";
   $result = pg_query($conn, $sql);
   while ($row = pg_fetch_row($result)) {
      $nome = $row[0];
   }
   return $nome;
}
//Mostra o curso selecionado na pág. Pesquisa por Curso
function mostraCurso($conn, $curso) {
   $sql = "select cur_nome from curso where cur_cod = $curso";
   $result = pg_query($conn, $sql);
   while ($row = pg_fetch_row($result)) {
      $nome = $row[0];
   }
   return $nome;
}
?>
```

APÊNDICE G – Arquivo inc_pesquisa.php: Página Pesquisa Professor.

```
<?php
$acao = $_REQUEST['acao'];
$prof = $_REQUEST['pro_cod'];
$timestamp = mktime(date("H") - 2, date("i"), date("s"), date("m"), date("d"), date("Y"), 0);
$dia_atual = gmdate("d/m/y H:i:s", $timestamp);
?>
<div class="centralizar">
   <div class="pesquisa">
      <div class="pesquisaB1">
         <h1>LOCALIZAÇÃO POR PROFESSOR</h1>
          [...]
                <form action="<?php echo $PHP_SELF; ?>" name="pesquisa" method="post">
                   <p><label for="pro_cod"> Selecione o professor</label></p>
                   <select name="pro_cod" id="pro_cod">
                      <?= listaProfessores($conexao); ?>
                   </select>
                   <input type="image" name="Localizar" id="Localizar" class="btnLocalizar"
src="imagens/pesquisa/btnLocalizar.png">
                   <input type="hidden" name="acao" value="LocalizarProfessor">
                </form>
          [...]
             <div class="boxLocalizacao">
                <h2>Localização</h2>
                <?php
                $sql = "select pre_numero, sal_numero, loc_hora
                      from professor p, localizacao l, sala s, predio e
                      where p.pro_cod = l.professor and
                      s.sal_cod = l.sala and
                      s.predio = e.pre_cod and  pro_cod = " . $prof . " and  loc_data = '" . $dia_atual . "'
                      order by loc_hora desc
                      LIMIT 4";
                $result = pg_query($conexao, $sql);
                $qnt = pg_num_rows($result);
                if (($acao == "LocalizarProfessor") && ($qnt != 0)) { ?>
                   <p>Última localização do(a) professor(a)
                    <?= mostraNome($conexao, $prof); ?> : </p>
                   <table border="0" class="tableLocal" align="center">
                      <thead>
                         <tr>
                            <th width="120">Localização</th>
                            <th width="50">Horário</th>
                         </tr>
                      </thead>
                      <tbody>
                         <?php while ($row = pg_fetch_row($result)) { ?>
                            <tr>
                               <td width="120">Prédio <?= $row[0]; ?> - Sala <?= $row[1]; ?></td>
                               <td width="50"><?= $row[2]; ?></td>
                            </tr>
                            <?php
                         }
                      } else {
                         if ($qnt == 0) {
                            ?>
                         <p>Professor(a) <?= mostraNome($conexao, $prof); ?> não esteve no Câmpus hoje.</p>
                      <?php } else { ?>
                         <p>Selecione o professor</p>
                    [...]
```

APÊNDICE H – Arquivo inc_pesquisa_predio.php: Página Pesquisa por Prédio.

```
[...]
    <form action="<?php echo $PHP_SELF; ?>" name="pesquisa" method="post">
      <p><label for="pre_cod"> Selecione o Prédio:</label></p>
      <select name="pre_cod" id="pre_cod">
        <?= listaPredios($conexao); ?>
      </select>
      <input type="image" name="Localizar" id="Localizar" class="btnLocalizar"
src="imagens/pesquisa/btnLocalizar.png">
      <input type="hidden" name="acao" value="LocalizarPorPredio">
    </form>
[...]
<div class="pesquisaPredioB2">
  <h1>Prédio <?= mostraNumPredio($conexao, $pred); ?></h1>
  <div class="boxResultado">
    <?php  $sql_cont = "select * from professor po, predio pe, localizacao l, sala s
        where s.sal_cod = l.sala and  l.loc_data = '" . $dia_atual . "' and  pre_cod = " . $pred;
    $rs = pg_query($conexao, $sql_cont);
    $qnt = pg_num_rows($rs);
    if ($qnt != 0) {
      if ($acao == "LocalizarPorPredio") { ?>
        <table width="400" border="0" align="center">
          <thead><tr>
              <th>Sala</th>
              <th>Professor</th>
              <th>Horário</th>
            </tr></thead>
          <tbody>
            <?php
            //Busca todos os professores
            $sql = "select pro_cod from professor";
            $result = pg_query($conexao, $sql);
            while ($row = pg_fetch_row($result)) { $professor = $row[0];
              //Seleciona o último check-in do professor  X na data atual
              $sql1 = "select max(loc_cod) from localizacao l, professor po, sala s
                where  po.pro_cod = l.professor and l.professor = '" . $professor . "' and
                loc_data = '" . $dia_atual . "'";
              $result1 = pg_query($conexao, $sql1);
              while ($row1 = pg_fetch_row($result1)) { $ultima_localizacao = $row1[0];
                //Busca a sala, professor e hora da última localização
                //conforme último check-in encontrado de cada professor
                $sql2 = "select sal_numero, pro_nome, loc_hora
                  from professor p, localizacao l, sala s, predio e
                  where p.pro_cod = l.professor and
                  s.sal_cod = l.sala and  s.predio = e.pre_cod and
                  pre_cod = " . $pred . " and  loc_cod = " . $ultima_localizacao;
                $result2 = pg_query($conexao, $sql2);
                while ($row2 = pg_fetch_row($result2)) { ?>
                  <tr>
                    <td width="75"><?= $row2[0]; ?></td>
                    <td width="250"><?= $row2[1]; ?></td>
                    <td width="75"><?= $row2[2]; ?></td>
                  </tr>
            <?php } } } ?>
          [...]
```

APÊNDICE I – Arquivo inc_pesquisa_curso.php: Página Pesquisa por Curso.

```
[...]
    <form action="<?php echo $PHP_SELF; ?>" name="pesquisa" method="post">
      <p><label for="cur_cod"> Selecione o Curso:</label></p>
      <select name="cur_cod" id="cur_cod">
        <?= listaCursos ($conexao); ?>
      </select>
      <input type="image" name="Localizar" id="Localizar" class="btnLocalizar"
src="imagens/pesquisa/btnLocalizar.png">
      <input type="hidden" name="acao" value="LocalizarPorCurso">
    </form>
  [...]
<h1>Curso: <?= mostraCurso($conexao, $curso); ?></h1>
<div class="boxResultado">
  <?php  $sql_cont = "select * from professor po, predio pe, localizacao l,
          sala s, curso c, professor_por_curso pc
            where s.sal_cod = l.sala and pe.pre_cod = s.predio and po.pro_cod = l.professor and
             l.loc_data = '" . $dia_atual . "' and po.pro_cod = pc.professor and
             pc.curso = c.cur_cod and pc.curso = ".$curso;
  $rs = pg_query($conexao, $sql_cont);
  $qnt = pg_num_rows($rs);
  if ($qnt != 0) {
    if ($acao == "LocalizarPorCurso") { ?>
      <table width="400" border="0" align="center">
        <thead><tr>
            <th>Prédio</th>
            <th>Sala</th>
            <th>Professor</th>
            <th>Horário</th>
          </tr></thead>
        <tbody>
          <?php
          //Busca todos os professores com vínculo em cursos
          $sql = "select pro_cod from professor p, professor_por_curso pc, curso c
              where c.cur_cod = pc.curso and pc.professor = p.pro_cod and
                  pc.curso = ".$curso;
          $result = pg_query($conexao, $sql);
          while ($row = pg_fetch_row($result)) { $professor = $row[0];
            //Seleciona o último check-in do professor  X na data atual
            $sql1 = "select max(loc_cod) from localizacao l, professor po
              where po.pro_cod = l.professor and  l.professor = '" . $professor . "' and
              loc_data = '" . $dia_atual . "'";
            $result1 = pg_query($conexao, $sql1);
            while ($row1 = pg_fetch_row($result1)) { $ultima_localizacao = $row1[0];
              //Busca o prédio, sala, professor e hora da última localização
              $sql2 = "select pre_numero, sal_numero, pro_nome, loc_hora
                from professor p, localizacao l, sala s, predio e
                where p.pro_cod = l.professor and s.sal_cod = l.sala and
                s.predio = e.pre_cod and loc_cod = " . $ultima_localizacao;
              $result2 = pg_query($conexao, $sql2);
              while ($row2 = pg_fetch_row($result2)) { ?>
                <tr>
                  <td width="75"><?= $row2[0]; ?></td>
                  <td width="75"><?= $row2[1]; ?></td>
                  <td width="250"><?= $row2[2]; ?></td>
                  <td width="75"><?= $row2[3]; ?></td>
                </tr>
          <?php } } } ?>
        [...]
```