

**UNIVERSIDADE FEDERAL DE SANTA MARIA
COLÉGIO TÉCNICO INDUSTRIAL DE SANTA MARIA
CURSO SUPERIOR DE TECNOLOGIA EM REDES DE
COMPUTADORES**

**SISTEMA DE MONITORAMENTO RESIDENCIAL
UTILIZANDO A PLATAFORMA ARDUINO**

TRABALHO DE CONCLUSÃO DE CURSO

Marcelo Marchesan

Santa Maria, RS, Brasil

2012

TCC/REDES DE COMPUTADORES/UFSM,RS

MARCHESAN, Marcelo

Tecnólogo 2012

SISTEMA DE MONITORAMENTO RESIDENCIAL UTILIZANDO A PLATAFORMA ARDUINO

Marcelo Marchesan

Trabalho de conclusão de curso apresentado ao Curso Superior de
Tecnologia em Redes de Computadores da Universidade Federal de
Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau
de
Tecnólogo em Redes de Computadores

Orientador: Prof. Celio Trois
Coorientador: Prof. Murilo Cervi

Santa Maria, RS, Brasil

2012

**UNIVERSIDADE FEDERAL DE SANTA MARIA
COLÉGIO TÉCNICO INDUSTRIAL DE SANTA MARIA
CURSO SUPERIOR DE TECNOLOGIA EM REDES DE
COMPUTADORES**

**A Comissão Examinadora, abaixo assinada, aprova
o Trabalho de Conclusão de Curso**

**SISTEMA DE MONITORAMENTO RESIDENCIAL UTILIZANDO A
PLATAFORMA ARDUINO**

elaborado por
Marcelo Marchesan

Como requisito parcial para a obtenção do grau de
Tecnólogo em Redes de Computadores

COMISSÃO EXAMINADORA

Celio Trois, Me.
(Presidente/Orientador)

Murilo Cervi, Dr.
(Coorientador)

Rogério Correa Turchetti, Me. (UFSM)

Tiago Antonio Rizzetti, Me. (UFSM)

Santa Maria, 06 de julho de 2012.

Algo só é impossível até que alguém duvide e acabe provando o contrário.

(Albert Einstein).

RESUMO

TRABALHO DE CONCLUSÃO DE CURSO
CURSO SUPERIOR DE TECNOLOGIA EM REDES DE
COMPUTADORES
UNIVERSIDADE FEDERAL DE SANTA MARIA

SISTEMA DE MONITORAMENTO RESIDENCIAL UTILIZANDO A PLATAFORMA ARDUINO

AUTOR: MARCELO MARCHESAN

ORIENTADOR: CELIO TROIS

Data e Local da Defesa: Santa Maria, 06 de julho de 2012.

Este trabalho objetiva estudar e desenvolver um sistema autônomo de gerenciamento de segurança residencial de baixo custo, confiável e eficiente. Para tanto, utilizou-se a plataforma Arduino, como o hardware que realiza o monitoramento de diversos sensores instalados na residência. O sistema de monitoramento residencial - SiMRe pode ser acessado através da Internet, na qual os usuários podem verificar as informações relativas ao sistema, além de poder customizar as configurações já existentes. Os testes realizados em uma maquete demonstraram a estabilidade do sistema, tornando-se um sistema seguro para ser implementado em residências. Seu custo também se mostrou acessível.

Palavras-Chave: Automação Residencial, Arduino, Segurança.

ABSTRACT

COMPLETION OF COURSE WORK
SUPERIOR COURSE OF TECHNOLOGY IN COMPUTER NETWORKS
FEDERAL UNIVERSITY OF SANTA MARIA

HOME MONITORING SYSTEM USING A PLATFORM ARDUINO

AUTHOR: MARCELO MARCHESAN

ADVISER: CELIO TROIS

Defense Place and Date: Santa Maria, July 6th, 2012.

This work aims to study and develop an autonomous residential security system. The system should have low cost, reliable and efficient. For this purpose, was used the Arduino platform to monitor several sensors installed on doors and windows. All the home monitoring system can be accessed through the Internet, where users can check all information relating to the system, and also customize these information. Tests conducted on a maquette demonstrated system stability, making it a secure system to be implemented in residences. Its cost was also accessible.

Keywords: Home Automation, Arduino, Security.

LISTA DE ILUSTRAÇÕES

Figura 1 - Projeto proposto.....	14
Figura 2 – Exemplo de um sinal analógico e um sinal digital.	20
Figura 3 - Funcionamento do sensor magnético.	21
Figura 4 - Quota de mercado para os principais servidores Web.....	23
Figura 5 - IDE da plataforma Arduino.....	31
Figura 6 - Arduino Mega 1280.....	32
Figura 7 – <i>Ethershield</i> baseado no microcontrolador ENC28J60.....	33
Figura 8 - Comunicação SPI, sentido da conexão.	36
Figura 9 - Jumpers entre o Arduino Mega 1280 e o ethershield ENC28J60.	36
Figura 10 - Adaptação para evitar conflito nas portas de E/S do Arduino.	37
Figura 11 - Ligação de dois sensores magnéticos ao Arduino.....	37
Figura 12 - Diagrama de estados - algoritmo Arduino.....	39
Figura 13 - Projeto conceitual do banco de dados.	42
Figura 14 - Modelo entidade relacionamento.	43
Figura 15 - Acesso ao Sistema Web.	45
Figura 16 - Histórico de eventos ocorridos.....	45
Figura 17. - Cadastro de Arduino.	46
Figura 18 - Mapa de monitoramento.	47
Figura 19 - Informações disponíveis em cada balão.	47
Figura 20 - Cadastro dos sensores	48
Figura 21 - Sistema de Monitoramento Residencial – SiMRe.	49
Figura 22 - Maquete do projeto proposto.	50

LISTA DE QUADROS

Quadro 1- Limites suportados pelo <i>PostgreSQL</i>	25
Quadro 2 - Comparação entre modelos da plataforma Arduino.....	30
Quadro 3 - Custos do projeto.	50

SUMÁRIO

1 INTRODUÇÃO	12
1.1 Objetivos	13
1.2 Justificativa.....	14
1.3 Trabalhos relacionados	14
1.4 Estrutura do trabalho	16
2 REVISÃO BIBLIOGRÁFICA	17
2.1 Microcontroladores	17
2.2 Sensores	19
2.2.1 Sensor magnético	20
2.2.2 Sensor de presença	21
2.2.3 Sensor de pressão	21
2.3 Desenvolvimento Web	22
2.3.1 Servidor de páginas Web	22
2.3.1.1 Apache	23
2.3.1.2 IIS – <i>Internet Information Services</i>	24
2.3.2 Sistema de gerenciamento de banco de dados – SGBD	24
2.3.2.1 <i>PostgreSQL</i>	25
2.3.2.2 <i>MySQL</i>	25
2.3.3 Linguagem de programação Web	26
2.3.3.1 ASP – <i>Active Server Pages</i>	26
2.3.3.2 PHP – <i>Hypertext Preprocessor</i>	26
3 MATERIAIS E MÉTODOS	28
3.1 Metodologia	28
3.2 Hardware	29
3.2.1 Plataforma Arduino.....	29
3.2.2 Estrutura do <i>hardware</i> do Arduino Mega 1280.....	32
3.2.3 Sensores	33
3.3 Sistema Web	34

4 RESULTADOS	35
4.1 Hardware	35
4.1.1 Fluxograma desenvolvido para o Arduino	38
4.2 Sistema Web	41
4.2.1 Modelagem do Banco de Dados	42
4.2.2 Desenvolvimento do sistema Web	44
4.3 Sistema de monitoramento residencial – SiMRe	48
5 CONSIDERAÇÕES FINAIS	52
REFERÊNCIAS	54
APÊNDICE	57

1 INTRODUÇÃO

A automação residencial é uma área que está em constante crescimento e tem enorme campo de atuação que vai desde a implementação de sensores para detectar fumaça, movimento ou pressão, até o acionamento automático de lâmpadas, portões eletrônicos e sirenes. Estes fatos foram constatados em uma pesquisa realizada pela *Motorola* (PESQUISA..., 2012).

Observou-se também o estudo feito em 2009 pelo Instituto Brasileiro de Geografia e Estatística (IBGE), no qual foi demonstrado que 47,2% da população brasileira com 10 anos ou mais de idade se sente insegura na cidade em que mora e 21,4% da população se sente insegura no domicílio em que reside (IBGE, 2010).

Essas porcentagens revelam uma tendência cada vez maior na busca por alternativas de segurança domiciliar, que vão desde métodos simples, como grades, olho mágico e correntes, até procedimentos mais sofisticados, como os citados por Pereira (2012): instalação de alarmes, câmeras de filmagem, seguranças particulares, entre outros.

No entanto, como afirma o *Jornal da Manhã* (CRIMES..., 2012), os recursos citados por Pereira (2012) muitas vezes têm custo alto (dependendo do sistema que será implantado) o que dificulta a aquisição desses serviços por pessoas de classe social mais baixa. Além disso, quando ocorre algum furto, arrombamento ou invasão na residência, muitos dos dispositivos de segurança mostram-se ineficazes, pois não informam ao proprietário em tempo real o fato ocorrido, nem o que realmente aconteceu.

Na tentativa de resolver esse impasse: segurança *versus* preço alto, o presente trabalho propõe a criação de um sistema de segurança residencial de baixo custo utilizando a rede mundial de computadores, visando minimizar eventuais problemas.

1.1 Objetivos

O presente trabalho tem como objetivo estudar e desenvolver um sistema autônomo de gerenciamento de segurança residencial de baixo custo, confiável e eficiente que possa ser acessado através da Internet.

Para tanto, os objetivos específicos são:

- a) Utilizar ferramentas livres, como *software* e *hardware open source*, para garantir o baixo custo do projeto;
- b) Possibilitar o uso de diversos dispositivos que confirmam maior segurança residencial, como por exemplo: sensores de presença, pressão, acionadores automáticos de lâmpadas, entre outros;
- c) Garantir o melhor funcionamento do sistema através da divisão de tarefas entre: monitoramento, envio de atividades, registro e armazenamento.
 - O monitoramento da residência será feito por uma rede de sensores, disposta em vários pontos da residência, como janelas e portas;
 - As alterações ocorridas nos sensores serão enviadas para o *hardware* que será o responsável por enviá-las ao sistema Web através da Internet;
 - O registro de todas as informações sobre anomalias ocorridas na residência ficarão por conta do sistema Web. Este será encarregado de realizar a interface entre o *hardware*, a rede de sensores e o usuário.

A *Figura 1* exemplifica o projeto proposto. Os números de um a oito representam o posicionamento dos sensores (localizados nas portas e janelas da residência); a letra “A” representa o local onde possivelmente deverá ser instalado o *hardware* responsável por monitorar os sensores.

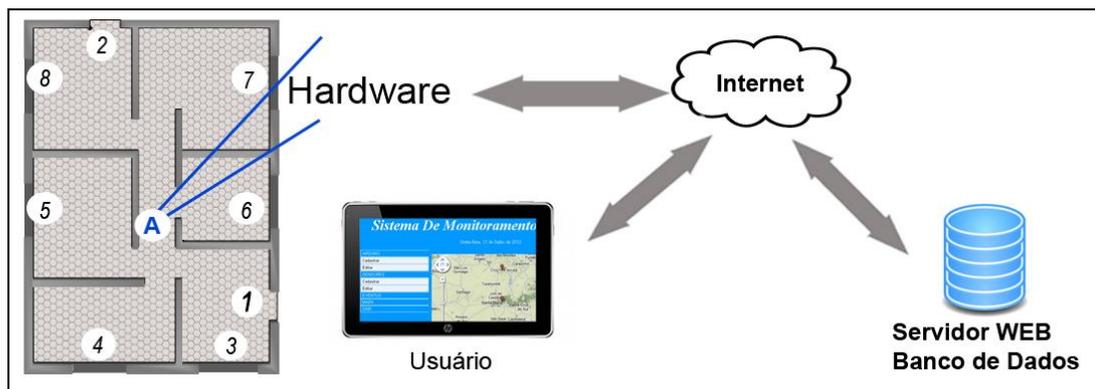


Figura 1 - Projeto proposto.

1.2 Justificativa

A justificativa do trabalho é feita por pelo menos quatro motivos: (i) por se tratar de um assunto relevante e, sobretudo, atual; (ii) pela necessidade de criação de um projeto de baixo custo e eficiente que garanta a segurança residencial; (iii) para trazer benefícios e segurança ao usuário, através das possibilidades de monitoramento residencial; e (iv) por interesse do pesquisador.

Ademais, no processo de elaboração desse trabalho serão aplicados os conhecimentos adquiridos durante o curso, o que por si só já o justificaria como relevante.

1.3 Trabalhos relacionados

Existem diversos trabalhos que tratam do monitoramento residencial e da automação, cujo desenvolvimento utiliza vários tipos de *hardware*. Todos eles procuram garantir maior conforto e segurança aos usuários. Dentre eles pode-se citar:

Marchette e Nunes (2011, p.1), cujo trabalho apresenta “a implementação de um protótipo integrando tecnologias de comunicação sem fio e redes TCP/IP para

aplicações em automação e monitoramento residencial” utilizando módulos ZigBee. Há também o trabalho de Pereira (2009), que propôs o desenvolvimento de uma

aplicação que proporcione a interação de um dispositivo móvel, ligado através de um protocolo de comunicação sem fio, com um computador servidor ou um sistema embarcado que se comunique diretamente com a central de alarme de uma residência [...], disponibilizando, remotamente, algumas funções de monitoramento do sistema de segurança da casa [...]. (PEREIRA, 2009, p.2)

Outro estudo relevante é o proposto por Barretta et al. (2009, p. 302), cujo trabalho procurou “apresentar uma arquitetura de monitoramento baseada em RSSFs [Rede de Sensores Sem Fio] para proteger ambientes domésticos”; e o trabalho de Baumann (2008, p.13), que objetivou “desenvolver um protótipo de um sistema de segurança residencial de monitoramento de imagens, sensores e acionamento de tomadas utilizando o *hardware Fox Board*, dotado do sistema operacional *Linux*”.

Com base nos trabalhos citados, procurou-se desenvolver um sistema de baixo custo, confiável e eficiente, que pudesse ser ajustado às necessidades de cada residência ou para qualquer outro tipo de edificação que necessite de um sistema de monitoramento.

Para tanto, pretende-se utilizar microcontroladores, portais Web, redes de comunicação e sensores de presença, de pressão e magnéticos. Serão estudadas, também, algumas plataformas de desenvolvimento disponíveis no mercado que atendam aos requisitos do projeto, como baixo custo, *hardware open source*, que poderá ser autônomo e controlado à distância, por meio da Internet, utilizando computadores e/ou aparelhos móveis, como celulares e *tablets*.

A existência de existirem diversos tipos de *hardware* que podem ser utilizados para garantir a segurança das residências, faz com que se procure opções que garantam maior segurança e que estejam há algum tempo no mercado, para que tenha maior suporte aos desenvolvedores, vasto referencial e menor custo ao usuário.

1.4 Estrutura do trabalho

O presente trabalho está estruturado da seguinte forma: no segundo capítulo serão apresentados alguns conceitos e definições importantes para o trabalho proposto.

No capítulo 3 são apresentadas as ferramentas escolhidas para desenvolver o sistema de monitoramento residencial e a metodologia de desenvolvimento do trabalho. Os resultados do projeto proposto já implementado, juntamente com a maquete do sistema serão apresentados no capítulo 4. Por fim, no capítulo 5 serão apresentadas as conclusões do trabalho, assim como algumas sugestões para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo serão abordados alguns conceitos e definições relevantes para o entendimento do sistema proposto. Primeiramente serão apresentados os *hardwares* pesquisados para realizar a implementação do projeto proposto: microcontroladores (seção 2.1); sensores (seção 2.2) e desenvolvimento Web (seção 2.3).

2.1 Microcontroladores

Os microcontroladores, por serem pequenos e apresentarem a melhor relação custo/benefício, estão presentes em quase tudo que envolve a eletrônica. Eles são dispositivos que possuem internamente todos os componentes necessários para seu funcionamento autônomo. Dito de outra forma, os

microcontroladores são computadores de propósito específico. Eles possuem tamanho reduzido, baixo custo e baixo consumo de energia. Devido a esses fatores há diversos segmentos, que os utilizam, tais como a indústria automobilística, de telecomunicações, de brinquedos, de eletrodomésticos, de eletroeletrônicos, bélica [...]. (SILVA, 2009, p.17)

Para o seu funcionamento autônomo, os microcontroladores incorporam vários recursos em uma única pastilha, tais como: memória, portas de entrada e saída, processador e diversas outras peças, todas montadas em um circuito integrado com pequenas dimensões. Veja a descrição completa dos componentes dos microcontroladores feita por Martins (2005, p. 16):

Tipicamente, um microcontrolador caracteriza-se por incorporar no mesmo encapsulamento um microprocessador (com a finalidade de interpretar as instruções de programa e processar dados), memória de programa (com a finalidade de armazenar de maneira permanente as instruções do programa), memória de dados (com a finalidade de memorizar os valores associados com as variáveis definidas no programa), uma série de pinos de entrada/saída (com a finalidade de realizar a comunicação do microcontrolador com o meio externo) e vários periféricos (tais como temporizadores, controladores de interrupção, temporizadores cão de guarda (WatchDog Timers – WDTs), comunicação serial, geradores de modulação por largura de pulso ou de PWM (Pulse Width Modulation),

conversores analógico/digital etc.), fazendo com que o hardware final fique extremamente complexo.

Martins (2005, p. 15) afirma ainda que existem vários tipos de microcontroladores. Segundo ele, o que os diferencia são: a velocidade do processamento; a quantidade de memória interna disponível para armazenar dados (memória de dados) e para armazenar as instruções de programas (memória de programa); “a quantidade de pinos de [...] [entrada e saída], a forma de alimentação, os tipos e as quantidades de periféricos, a arquitetura e o conjunto de instruções disponibilizado nos circuitos internos”.

Um exemplo de microcontrolador, talvez o mais popular, é o da série PIC, fabricado pela empresa *Microchip Technology Inc.*, que se popularizou pelo bom plano de marketing “baseado na disseminação de uma ferramenta de auxílio à construção de programas – o MPLAB IDE. [...] [e, por possuírem] uma linguagem Assembly menos complexa em relação àquelas disponibilizadas por outros fabricantes” (MARTINS, 2005, p. 15). Esses microcontroladores da série PIC têm várias opções de memória de programas, como o *One Time Programmable* (OTP) e o *Erasable and Programmable Read Only Memory* (EPROM). Ademais, têm “opções de baixa tensão e inúmeros tipos de circuito osciladores, assim como várias opções de encapsulamento” (MARTINS, 2005, p. 17).

Outro microcontrolador fortemente citado na literatura e que tem ampla aceitação no mercado é o da família AVR, fabricado pela empresa ATMEL. Este microcontrolador apresenta “bom desempenho frente ao número de instruções executadas por ciclo de *clock*.” (LIMA, SCHWARZ, 2009, p. 94). Além disso, Lima e Schwarz (2009, p. 94) afirmam que a ATMEL fornece um programa gratuito, chamado AVR Studio, para a programação desses microcontroladores, que podem ser programadas tanto em linguagem Assembly, quanto em linguagem C (empregando-se o compilador C gratuito).

Após a análise desses dois tipos de microcontroladores, estudou-se a plataforma Arduino, que se caracteriza por utilizar um microcontrolador da família AVR. “O que antes necessitava de conhecimentos técnicos específicos de eletrônica e programação, agora se tornou extremamente simples e até intuitivo.” (CARVALHO, 2011, p. 34). Além disso, essa plataforma facilita o uso de microcontroladores. Com ela, pode-se monitorar sensores, pode-se comunicar com computadores e celulares

e, inclusive, pode-se controlar algumas funções como ligar e desligar lâmpadas (através do controle de relés que funcionam como interruptores), abertura de fechaduras elétricas, leitura de sensores, etc.

Adicionado a tudo isso, tem-se que

O Arduino oferece uma interface de *hardware* proporcionando todo o circuito necessário para funcionamento do microcontrolador e uma interface e ambiente de desenvolvimento em *software* para programação. Por ser uma plataforma de código aberto (*open-source*) há uma grande comunidade de desenvolvedores do mundo inteiro que publicam bibliotecas já com toda a programação pronta para se usar, com funções específicas, como, por exemplo, o controle de servo motores ou leitura de sensores analógicos. (CARVALHO, 2011, p. 34)

2.2 Sensores

Sensores são dispositivos que mudam de estado conforme a interação com o ambiente. Seu *hardware* pode ser composto por diversos componentes eletrônicos ou por apenas um componente. Estes dispositivos podem interagir com diversos tipos de grandezas físicas, tais como temperatura, movimento, pressão, entre outras, convertendo essas grandezas em sinais elétricos analógicos ou digitais.

Os que possuem saídas analógicas tem seu valor oscilando conforme a variação da grandeza de entrada. Já os com saída digital, geralmente digital binário, assumem apenas dois valores bem definidos. Este comparativo entre sinais analógicos e digitais pode ser melhor visualizado na *Figura 2*.

Dentre os vários tipos de sensores, serão apresentados, no decorrer das próximas seções, os sensores considerados mais importantes para a proteção de uma residência: magnético, de presença e de pressão.

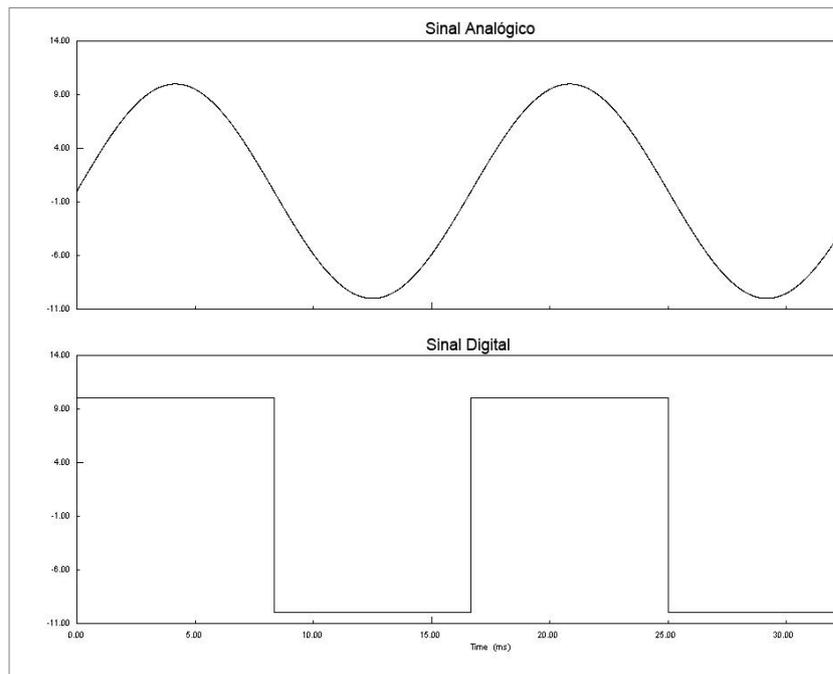


Figura 2 – Exemplo de um sinal analógico e um sinal digital.

2.2.1 Sensor magnético

Os sensores magnéticos são formados por duas placas ferromagnéticas separadas e encapsuladas por um vidro. Quando um campo magnético (ímã) é aproximado do sensor, suas chapas são alinhadas ao campo externo, estabelecendo contato elétrico (*Figura 3.b*) e possibilitando a passagem de corrente. Quando o campo magnético é afastado, as chapas voltam a se afastar, eliminando o contato, e o sensor muda seu estado para aberto (*Figura 3.a*).

Este tipo de sensor magnético é considerado o mais simples e é composto por duas partes: o sensor e o ímã. Há várias possíveis aplicações para o sensor magnético. Ele pode ser utilizado para monitorar objetos, por exemplo. Nesse caso, o ímã fica preso a um objeto de referência, enquanto o sensor fica localizado próximo a ele para que possam interagir.

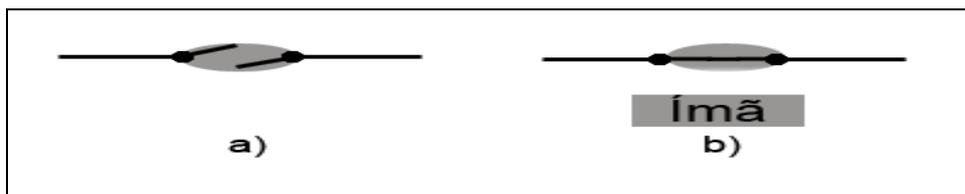


Figura 3 - Funcionamento do sensor magnético.

Existem outros tipos sensores magnéticos que são utilizados em várias aplicações, tais como: medir correntes elétricas a partir do campo magnético originado por elas, bússola digital através da análise do campo magnético do planeta Terra, entre outras.

2.2.2 Sensor de presença

O sensor de presença ou de infravermelho detecta movimento de objetos em seu raio de atuação através do calor emitido pelo objeto. Este tipo de sensor é sensível à mudança de radiação infravermelha, por isso, quando uma pessoa fica imóvel no seu raio de atuação o mesmo não detecta sua presença, pelo fato de não detectar essa variação da radiação. Possui saída digital, tendo somente dois estados: detecção de movimento e sem detecção.

2.2.3 Sensor de pressão

Baseado num condutor ou semiconductor, este tipo de sensor tem sua resistência alterada conforme a deformação que é aplicada sobre ele. Assim, a tensão elétrica da saída analógica varia conforme a pressão aplicada.

2.3 Desenvolvimento Web

Esta seção apresenta diversos aplicativos que foram estudados a fim de desenvolver a interface Web do projeto proposto. Ela tem por finalidade armazenar diversas informações que poderão ser acessadas de qualquer lugar através da Internet.

2.3.1 Servidor de páginas Web

Alecrim (2006 apud KLABUNDE, 2007, p. 28) esclarece que um servidor Web é um computador que processa solicitações *Hyper Text Transfer Protocol* (HTTP). Um sistema Web é baseado na arquitetura cliente/servidor. Conforme Tanenbaum (2003), nesta arquitetura, o cliente envia requisições ao servidor Web e aguarda uma resposta. Enquanto isso, o servidor Web é responsável por processar as requisições recebidas e enviar os resultados para o cliente.

Os servidores Web suportam diversos tipos de linguagens de programação, estáticas como o HTML e dinâmicas como PHP¹ ou ASP².

Uma pesquisa realizada pela Netcraft (JANUARY..., 2012) quantificou a utilização de diversos tipos de servidores Web disponíveis no mercado no período de outubro de 1995 a janeiro de 2012. Esses dados são apresentados na *Figura 4*.

Pelos dados, percebe-se que houve um crescente fluxo de utilização do servidor Apache (azul) desde outubro de 1995, com uma ligeira queda de 2006 a 2009. Mesmo assim, esse servidor é o mais utilizado desde 1995. Outro servidor que vinha ganhando destaque de uso é o da Microsoft (linha vermelha) que, teve um grande aumento justamente no período de queda da Apache (2006 a 2008). Ao contrário, o servidor MCSA teve queda brutal desde a mesma época.

¹ *Hypertext Preprocessor.*

² *Active Server Pages.*

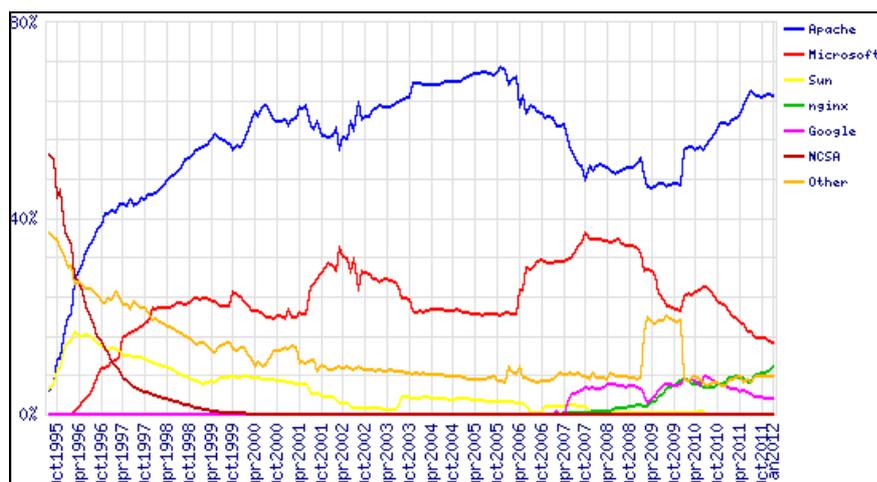


Figura 4 - Quota de mercado para os principais servidores Web.
 Fonte: Netcraft (JANUARY..., 2012).

A seguir serão descritos os dois aplicativos mais utilizados: 1) O Apache e 2) *Internet Information Services*, construído pela *Microsoft*.

2.3.1.1 Apache

Segundo informações do projeto Apache (ABOUT..., 2012), o apache é uma aplicação desenvolvida que utiliza o conceito de *software open source*. Este conceito visa o desenvolvimento de *software* sem fins lucrativos e mantidos por uma comunidade de desenvolvedores.

Abaixo seguem algumas características que tornaram este servidor de páginas Web muito atrativo:

- a) Suporte para plataformas *open source* e proprietárias;
- b) Sem custos para sua utilização;
- c) Estruturação em módulos;
- d) Suporte a várias linguagens: *PHP, PERL, HTML, PYTHON*;
- e) Controle de acesso e encriptação utilizando certificados digitais, *SSL*, entre outros.

2.3.1.2 IIS – *Internet Information Services*

A solução proprietária IIS (do inglês, *Internet Information Services*) oferecida pela *Microsoft* para servidores Web é disponibilizada somente para a plataforma *Windows*. Abaixo seguem algumas características:

- a) Código proprietário;
- b) Suportada somente na plataforma *Windows*;
- c) Suporte a várias linguagens: *ASP*, *PHP*, *PERL*, *HTML*;
- d) Controle de acesso e encriptação, utilizando certificados digitais, *SSL*, entre outros.

2.3.2 Sistema de gerenciamento de banco de dados – SGBD

A utilização de banco de dados está presente no cotidiano da sociedade. Eles são utilizados em formas simples, como uma agenda que armazena nomes, endereços e telefones de diversas pessoas; e, também, em sistemas mais complexos, como nos sistemas de saque de dinheiro em caixas eletrônicos. Neste último caso, há um sistema que armazena as informações da conta e atualiza o saldo.

Um sistema de gerenciamento de banco de dados (SGBD) pode ser definido como

uma coleção de programas que permite aos usuários criar e manter um banco de dados. O SGBD é, portanto, um sistema de software de propósito geral que facilita os processos de definição, construção, manipulação e compartilhamento de bancos de dados entre vários usuários e aplicações. A **definição** de um banco de dados implica especificar os tipos de dados, as estruturas e as restrições para os dados a serem armazenados em um banco de dados. (ELMASRI, 2005, p. 3, grifo do autor)

Assim, pela sua funcionalidade e praticidade, há no mercado uma grande quantidade de SGBD's disponíveis, tais como *PostgreSQL* e *MySQL*, objetos de estudos das próximas seções.

2.3.2.1 PostgreSQL

PostgreSQL (SOBRE..., 2012) é um SGDB objeto-relacional de código aberto, que é suportado por diferentes sistemas operacionais, como Microsoft Windows, distribuições *Linux*, *Solaris*, *Mac OS*, entre outras. Seu projeto foi iniciado há mais de quinze anos, e está em constante evolução, pois é mantido por uma comunidade de colaboradores.

O *PostgreSQL* (SOBRE..., 2012) ainda informa que a aplicação “é altamente escalável, tanto na quantidade enorme de dados que pode gerenciar, quanto no número de usuários concorrentes que pode acomodar”. No Quadro 1, abaixo, é possível analisar alguns limites suportados por este SGDB.

Tamanho Máximo do Banco de Dados	Ilimitado
Tamanho máximo de uma Tabela	32 TB
Tamanho Máximo de uma Linha	1.6 TB
Tamanho Máximo de um Campo	1 GB
Máximo de Linhas por Tabela	Ilimitado
Máximo de Colunas por Tabela	250–1600 dependendo do tipo de coluna
Máximo de Índices por Tabela	Ilimitado

Quadro 1- Limites suportados pelo *PostgreSQL*.

Fonte: *PostgreSQL* (SOBRE..., 2012).

2.3.2.2 MySQL

Alecrim (2008), afirma que “O *MySQL* é um dos sistemas de gerenciamento de banco de dados mais populares que existe e, por ser otimizado para aplicações Web, é amplamente utilizado na Internet”. Ainda segundo Alecrim (2008), esta aplicação é muito utilizada em conjunto com a linguagem de programação Web

PHP, disponível em vários sites de hospedagem de páginas, pois tem um bom desempenho em conjunto.

Sua aplicação é suportada por diferentes sistemas operacionais, entre eles podemos citar *Windows* e o *Linux*. Este SGBD é desenvolvido utilizando o princípio do código livre, mas ao contrário de muitos aplicativos *open source*, possui também uma licença comercial, isto é, uma versão paga, que garante suporte diferenciado dos desenvolvedores.

2.3.3 Linguagem de programação Web

As linguagens de programação Web são utilizadas para desenvolver sistemas que serão acessados através de uma rede, podendo ser acesso local (intranet) ou acesso remoto (extranet, Internet).

2.3.3.1 ASP – *Active Server Pages*

O ambiente de programação ASP, cria *scripts* para gerar páginas dinâmicas que foi desenvolvido pela *Microsoft*, por tanto, é proprietário. Foi desenvolvido utilizando a linguagem de programação *Visual Basic*. Os *scripts* são executados do lado do servidor, gerando para o cliente, páginas em HTML puro, onde qualquer navegador Web possa acessar.

Tem suporte ao ASP, os servidores de página Web IIS, desenvolvido pela *Microsoft*, Apache, entre outros.

2.3.3.2 PHP – *Hypertext Preprocessor*

A linguagem de programação Web PHP foi desenvolvida em 1994 por Rasmus Lerdorf. Essa linguagem utiliza um conjunto de *scripts* e é

voltada para aplicações Web, embutido no HTML. O código é delimitado por *tags* iniciais e finais, que permitem ao programador oscilar entre o HTML e o PHP. A maneira como o PHP é executado diferencia-se do *Javascript*, pois é do lado do cliente que o código é executado no servidor, gerando um HTML e o cliente acaba recebendo os resultados gerados pelos scripts. Possui código aberto, não se necessita da compra de licença, o programa é gratuito, é multiplataforma, tem acesso a banco de dados e faz o processamento de imagens ao enviá-las para o navegador do usuário. (HACKENHAAR; CARDOSO, 2010)

A utilização do PHP tem grande vantagem, pois com ela pode-se criar páginas com conteúdo dinâmico, sem a necessidade de alterar os códigos fonte do sistema, através da integração com um banco de dados que registre as informações que serão atualizadas.

Algumas aplicações que suportam a linguagem de programação PHP:

- Servidores de páginas Web: Apache, IIS, entre outros;
- Sistemas operacionais: distribuições *Linux*, *Microsoft Windows*, entre outros.
- Suporte nativo ao banco de dados: *MySQL*, *Oracle*, *Interbase*, entre outros;

Além disso, esta linguagem é concebida pelo conceito de código livre, sendo constantemente atualizada e com grande quantidade de documentação disponível para consultas.

3 MATERIAIS E MÉTODOS

Neste capítulo será abordada a metodologia utilizada para a realização do projeto proposto. A descrição geral dos procedimentos metodológicos percorridos durante a execução do trabalho estão na seção 3.1. O *hardware* utilizado, bem como suas características de funcionamento, são apresentados na seção 3.2. Na seção 3.3 são apresentadas as ferramentas e tecnologias escolhidas para o desenvolvimento do sistema *Web*.

3.1 Metodologia

A metodologia utilizada para o desenvolvimento deste trabalho se deu em quatro fases/etapas principais. Inicialmente, fez-se uma revisão bibliográfica sobre todas as tecnologias e equipamentos que poderiam ser empregadas na execução deste projeto.

A partir desse estudo, escolheu-se as tecnologias:

- plataforma de desenvolvimento Arduino;
- Sensores magnéticos;
- SGBD: *MySQL*;
- Linguagem de programação Web: PHP.

Após esta primeira fase, foram adquiridos: protótipo de hardware e sensores magnéticos, conforme serão descritos nas seções seguintes.

Na segunda etapa, foram desenvolvidos os algoritmos necessários para o funcionamento do protótipo de *hardware*, seguindo as especificações do projeto proposto.

Em seguida, desenvolveu-se um sistema Web, visando coletar eventos ocorridos no protótipo. Esse sistema permitirá a visualização do histórico de eventos e também possibilitará algumas interações do usuário com o sistema de segurança.

Na quarta e última etapa do trabalho, fase de aplicação, agregando-se estas partes, criou-se o Sistema de Monitoramento Residencial (SiMRe). Instalou-se o sistema em uma maquete para verificar o funcionamento do mesmo.

3.2 Hardware

Os microcontroladores são comercializados na sua maioria sem nenhuma plataforma de desenvolvimento, isto é, somente o microcontrolador. Sendo necessário para sua utilização, realizar a integração com outros componentes: como diodos, reguladores de tensão e resistores. Só assim, conseguiria-se implementar o projeto proposto. No entanto, como um dos objetivos deste estudo é elaborar um sistema de gerenciamento de segurança residencial de baixo custo, a opção por microcontroladores sem nenhuma plataforma de desenvolvimento tornaria o preço final do produto muito caro.

Tendo em vista isso, escolheu-se a plataforma de desenvolvimento Arduino para a realização do desenvolvimento do protótipo, pois esta plataforma já possui todos esses componentes integrados em uma placa de circuito impresso. A facilidade de interação com sensores digitais e analógicos oferecida pela plataforma também foi significativa no momento da escolha. As características dessa plataforma serão apresentadas a seguir:

3.2.1 Plataforma Arduino

A plataforma de desenvolvimento Arduino possui diversas características que a tornaram uma solução atraente. Por oportuno, algumas dessas características:

- Regulador de tensão de entrada, podendo utilizar fontes de alimentação que possuam tensão entre 6 a 20 *volts*;
- Saída de alimentação de 3,3 e 5 volts para ligar outros componentes ao *hardware* sem ter necessidade de utilizar outra fonte de alimentação;

- Conectores em todas as portas de entrada e saída do microcontrolador facilitando a integração com outros componentes, como por exemplo, sensores de movimento;
- Ambiente de desenvolvimento de algoritmos próprio e *upload* através da porta USB de qualquer computador.

Segundo informações do site, Arduino (ARDUINO..., 2012) “é uma plataforma *open-source* de prototipagem eletrônica baseada na flexibilidade, *hardware* e *software* fácil de usar”. Ela foi desenvolvida para ser totalmente autônoma, isto é, necessitar somente de uma fonte de alimentação para executar suas rotinas.

Por utilizar os microcontroladores da família *AVR* (cf. seção 2.1), essa plataforma é composta por inúmeras versões. Algumas listadas no quadro abaixo:

	Arduino UNO	Arduino Mega 2560	Arduino Mega 1280
Microcontrolador	ATmega 328	ATmega2560	ATmega1280
Nº de portas de E/S³ digital	14 (sendo que 6 podem ser usadas como PWM ⁴)	54 (sendo que 15 podem ser usadas como PWM)	54 (sendo que 14 podem ser usadas como PWM)
Nº de portas de Entrada analógica	6	16	16
Flash memory	32KB	256KB	128KB
Clock speed	16 MHz	16MHz	16MHz
Bootloader	0.5KB	8KB	4KB

Quadro 2 - Comparação entre modelos da plataforma Arduino.

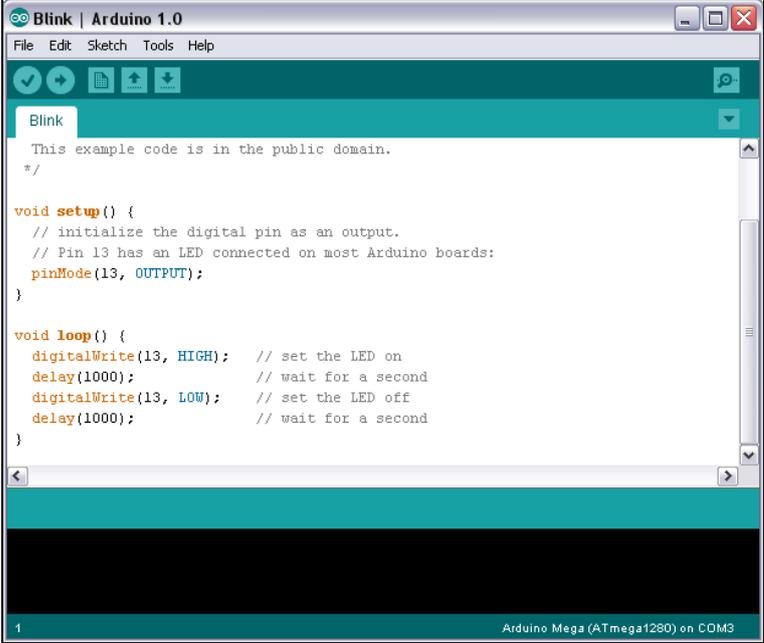
Fonte: Arduino (ARDUINO MEGA; ARDUINO UNO; ARDUINO MEGA 2560, 2012).

³ Entrada e saída.

⁴ A modulação por largura de pulso (do inglês, *Pulse-Width Modulation*) consiste em transportar informação ou controlar a potência de outro circuito, através da manipulação da razão cíclica do sinal. A razão cíclica é o tempo em que o sinal permanece alto em relação ao período de cada pulso de *clock*.

Como se pode observar, com a variação dos microcontroladores utilizados, obtém-se uma plataforma muito versátil em relação à quantidade de portas de E/S e memória para a gravação dos algoritmos.

A plataforma Arduino possui Interface de Desenvolvimento - IDE (*Figura 5*) multiplataforma, isto é, há a possibilidade de utilizar a IDE com sistemas operacionais distintos, como, *Windows, Linux, Mac OS*. Esta característica facilita o desenvolvimento dos algoritmos, podendo ser escritos, alterados e enviados ao Arduino de qualquer Sistema Operacional suportado.

The image shows a screenshot of the Arduino IDE interface. The title bar reads "Blink | Arduino 1.0". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for saving, opening, and running. The main text area contains the following code:

```
Blink
This example code is in the public domain.
*/

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);           // wait for a second
  digitalWrite(13, LOW); // set the LED off
  delay(1000);           // wait for a second
}
```

The status bar at the bottom indicates "1" on the left and "Arduino Mega (ATmega1280) on COM3" on the right.

Figura 5 - IDE da plataforma Arduino.

O modelo escolhido para realizar o desenvolvimento do protótipo foi o Arduino Mega 1280, pois possui uma quantidade aceitável de memória para o armazenamento de códigos e um número razoável de portas de entrada e saída para conexão dos sensores. A seguir, serão apresentadas mais informações sobre as características do Arduino Mega 1280.

3.2.2 Estrutura do *hardware* do Arduino Mega 1280

A plataforma Arduino Mega 1280 possui pequenas dimensões (10 cm x 5,25 cm), podendo ser instalado em qualquer local na residência. Na *Figura 6*, é possível fazer a análise das diferentes partes que compõem seu hardware.

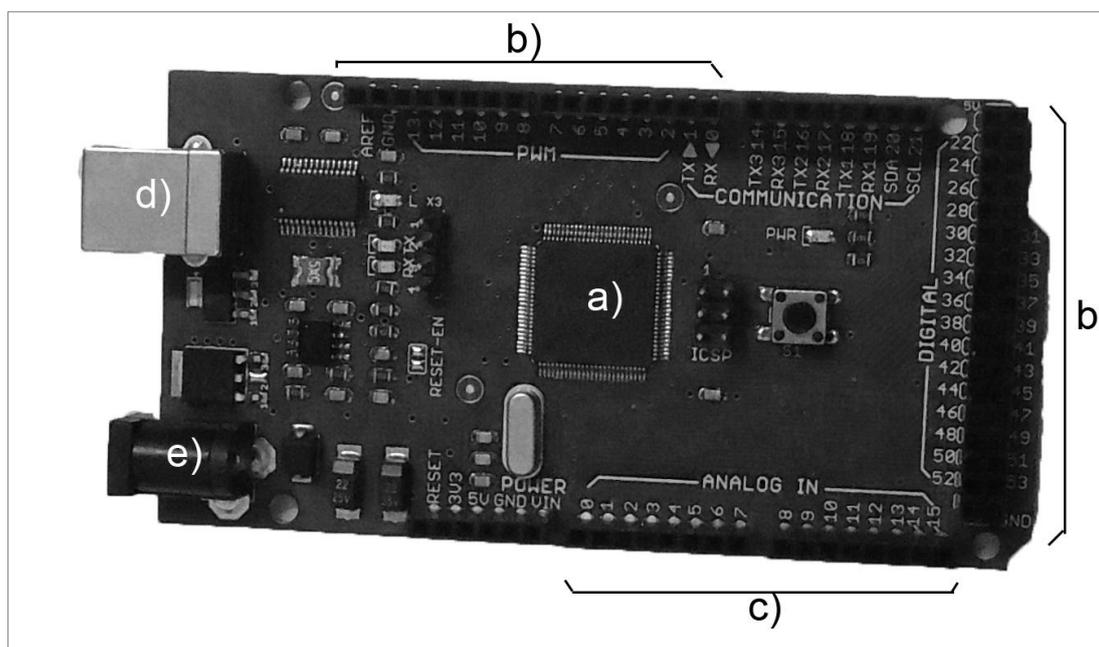


Figura 6 - Arduino Mega 1280.

As características do Arduino Mega 1280 são as seguintes:

- a) Microcontrolador ATmega1280;
- b) 54 portas digitais de E/S (sendo que 14 delas podem ser utilizadas como saída PWM);
- c) 16 portas de entrada analógicas (podem ler valores analógicos de tensão e convertê-los em valores binários correspondentes, de 0 a 1023);
- d) Conexão padrão USB B;
- e) Conector fêmea de 2,1mm para ligar fonte de alimentação;

Além desses recursos de hardware, a plataforma Arduino conta com inúmeros acessórios próprios, denominados *Shields*. Os *Shields* são placas de circuito impresso desenvolvidos para realizar outras tarefas, como por exemplo, o *ethershield* baseado no microcontrolador *ENC28J60* (Figura 7), que tem por finalidade realizar a conexão do Arduino com uma rede ethernet. Esses *shields* foram projetados na sua maioria para “encaixar” sobre o Arduino, tendo em vista a redução do espaço ocupado e utilização da mesma fonte de alimentação.

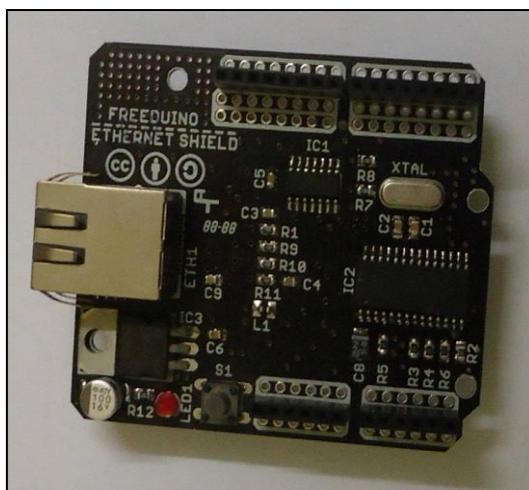


Figura 7 – *Ethershield* baseado no microcontrolador *ENC28J60*.

3.2.3 Sensores

Os sensores utilizados no protótipo foram somente do tipo magnético (descrito na seção 2.2.1), porém é importante ressaltar que poderá ser utilizado qualquer tipo de sensor que possua saída. Posicionou-se o ímã junto à porta, enquanto o sensor foi preso ao portal, assim, quando a porta estiver fechada o circuito também estará, possibilitando a passagem de corrente elétrica. Caso a porta seja aberta, ocorrerá a mudança de estado do sensor, pois com o afastamento do ímã e a passagem de corrente elétrica é cessada.

3.3 Sistema Web

Para o desenvolvimento do sistema Web foram analisadas algumas soluções proprietárias e outras *open source*. Como o objetivo é obter um sistema robusto e com o menor custo possível, optou-se pelas seguintes soluções de código livre:

- ✓ Como servidor Web, foi escolhido o servidor Apache, por se tratar de uma aplicação *open source*, com vasta documentação disponível;
- ✓ A linguagem de programação escolhida foi o PHP, devido ao conhecimento prévio do autor o que facilitará a implementação do sistema Web;
- ✓ O sistema de gerenciamento de banco de dados será o *MySQL* conforme descrito na seção 2.3.2, por se tratar de um sistema de banco de dados otimizado para aplicações Web.

4 RESULTADOS

Neste capítulo são apresentados os resultados do projeto proposto. Sendo assim, na primeira seção será apresentado o resultado da utilização do *hardware* (plataforma Arduino, *ethershield* e sensores). Em seguida, o sistema Web desenvolvido e, por último, a integração das partes formando o SiMRe.

4.1 Hardware

O Arduino Mega 1280 não tem o posicionamento padrão dos pinos como os demais modelos (Uno, Duemilanove, Diecimila). Assim, os pinos responsáveis pela comunicação SPI (*Serial Peripheral Interface*) foram trocados de lugar, causando a incompatibilidade de alguns *shields*, como é o caso do *ethershield* e algumas bibliotecas que utilizam uma conexão SPI para comunicação.

Conforme Monaro (2007, p. 11), “o barramento SPI é uma interface de dados serial síncrona padronizada pela Motorola que funciona em modo *full-duplex*⁵. Os equipamentos se comunicam em modo mestre/escravo, onde o equipamento mestre inicializa a comunicação”.

A comunicação SPI é composta por quatro canais (*Figura 8*):

- MISO (*Master In Slave Out*): dados enviados do *Slave* para o *Master*;
- MOSI (*Master Out Slave In*): dados enviados do *Master* para o *Slave*;
- SCK (*Serial Clock*): pulsos gerados pelo *Master* para sincronizar a transmissão;
- SS (*Slave Select*): utilizado para ativar/desativar a comunicação com um *Slave*;

⁵ Neste tipo de configuração é possível transmitir e receber informações simultaneamente.

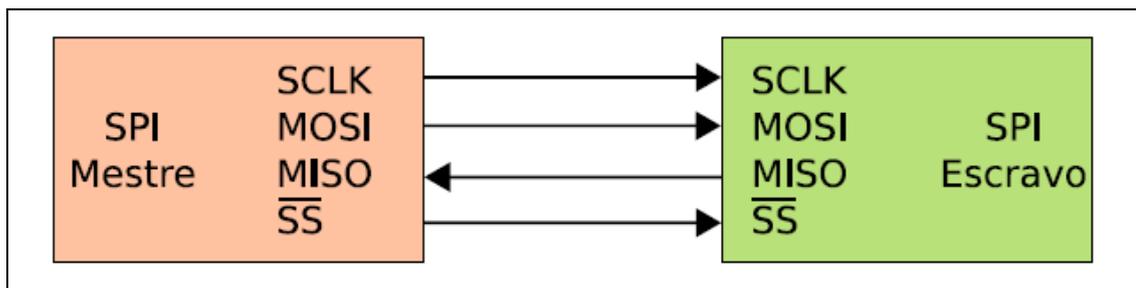


Figura 8 - Comunicação SPI, sentido da conexão.
 Fonte: MONARO (2007, p. 12).

Para estabelecer a comunicação entre o Arduino Mega 1280 e o ethershield, foi necessário inserir quatro *jumpers* (conforme *Figura 9*) para que fosse possível deixá-los “encaixados”, já que os pinos responsáveis pela conexão SPI estão dispostos em locais diferentes. Também foi preciso adaptar o encaixe entre ambos para que não ocorresse conflito entre as portas E/S do Arduino que ficam embaixo dos pinos do SPI do *ethershield* (como mostra a *Figura 10*).

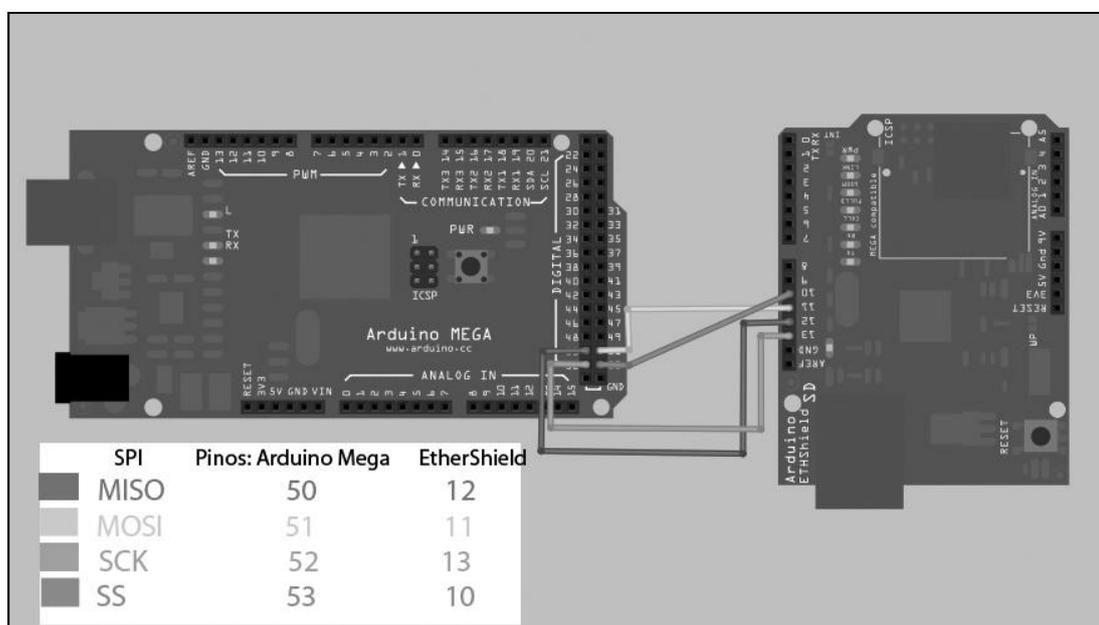


Figura 9 - *Jumpers* entre o Arduino Mega 1280 e o ethershield ENC28J60.

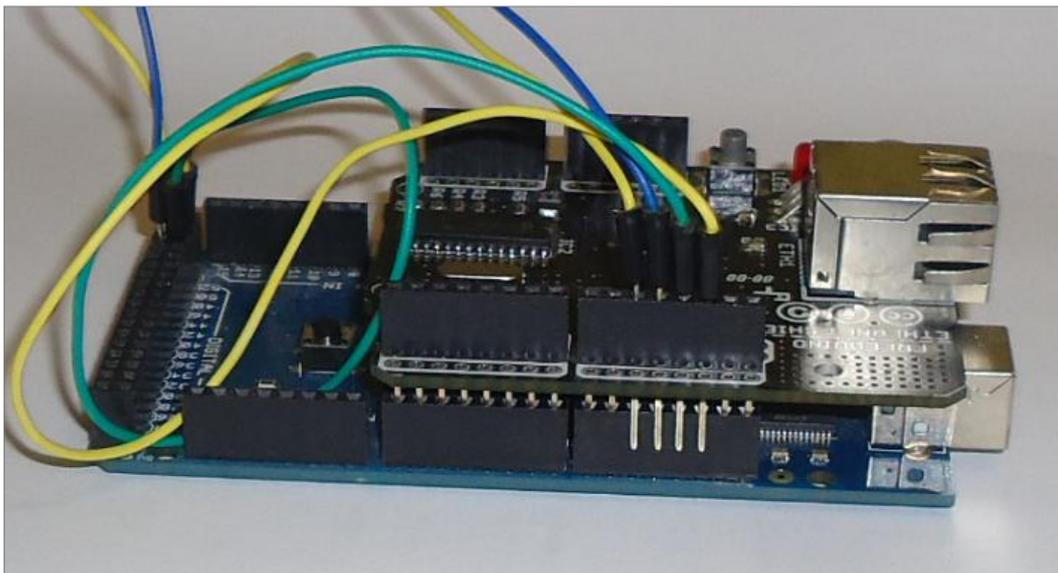


Figura 10 - Adaptação para evitar conflito nas portas de E/S do Arduino.

Com essas alterações, o conjunto de hardware (Arduino + ethershield) foi possível estabelecer a comunicação com a rede *ethernet*.

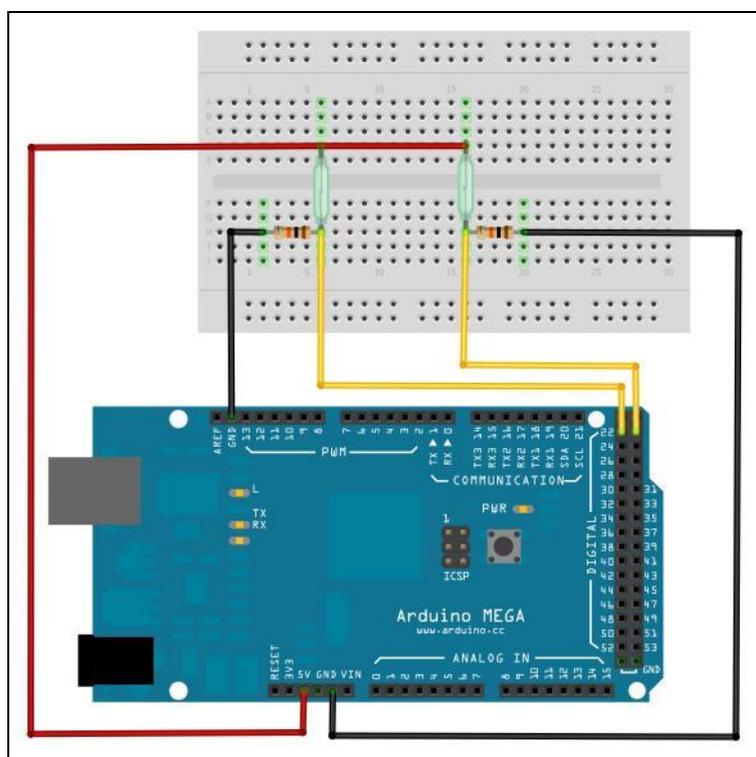


Figura 11 - Ligação de dois sensores magnéticos ao Arduino.

Para conectar os sensores ao Arduino foi necessário utilizar um resistor no circuito. A

Figura 11 mostra a ligação de dois sensores magnéticos ao Arduino, utilizando resistores em modo *pull-down*⁶, que tem por objetivo manter as portas de entrada (pinos 22 e 24 utilizados no exemplo) em valores lógicos próximos a zero.

4.1.1 Fluxograma desenvolvido para o Arduino

O Arduino precisa realizar um monitoramento constante de todos os sensores que estão conectados a ele. Quando um sensor mudar seu estado, o Arduino deve tomar uma decisão sobre qual procedimento irá tomar sobre aquele evento. No entanto, para que isso ocorra é necessário desenvolver um algoritmo capaz de interagir com a mudança de estado dos sensores e os demais dispositivos ligados a ele.

O algoritmo desenvolvido pode ser melhor visualizado no diagrama de estados apresentado na *Figura 12*.

Para uma melhor compreensão do diagrama de estados, deve-se saber:

- Estado HIGH ou fechado ou desativado: quando o sensor está próximo ao ímã (porta ou janela fechada), possibilitando a passagem de uma tensão elétrica, este é o estado padrão do sensor no sistema de monitoramento residencial;
- Estado LOW ou aberto ou ativado: quando o sensor está afastado do ímã (porta ou janela aberta), impossibilitando a passagem de uma tensão elétrica;
- Estado atual: é o estado real do sensor naquele exato momento;
- Cada porta de E/S possui somente um sensor conectado;
- Lista de sensores: quando um sensor é cadastrado na lista, o mesmo só é removido quando o sensor voltar para o estado fechado, este procedimento é necessário, para não ocorrer várias notificações sobre um mesmo evento.

⁶ A utilização de resistores em modo *pull-down* garante que a tensão na porta de entrada não oscile, mantendo-se próximo à zero, quando o sensor estiver aberto. Isso evita que a tensão fique flutuando e impede que o Arduino interprete erroneamente que o sensor está fechado.

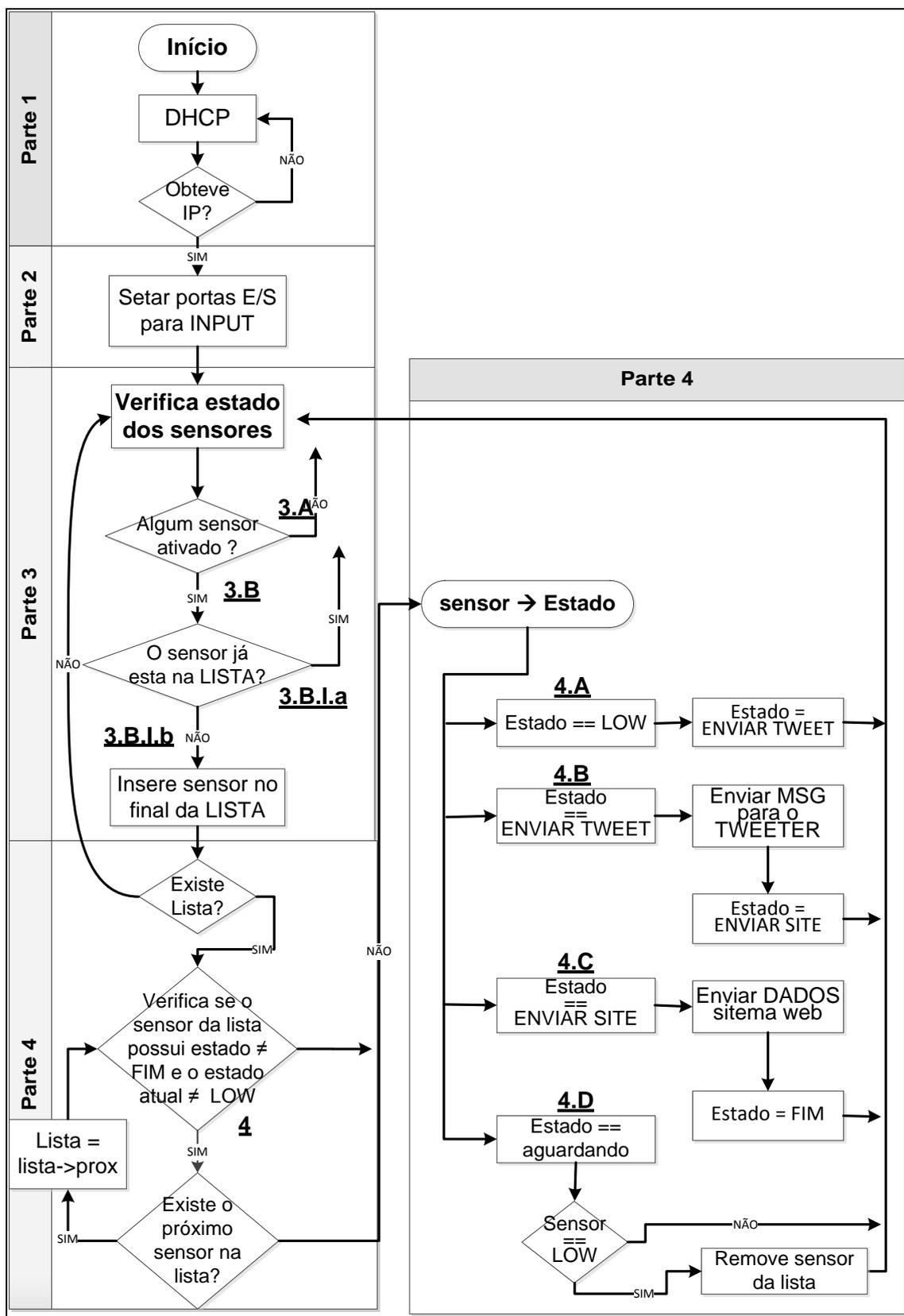


Figura 12 - Diagrama de estados - algoritmo Arduino.

O Arduino deve estar conectado a uma fonte de alimentação, possuir uma conexão com a Internet e seus sensores devem estar conectados nas devidas portas. Com esses requisitos “preenchidos”, o diagrama de estado pode ser dividido em 4 partes:

1. Inicialmente o Arduino realiza uma requisição de DHCP através do *ethershield* para rede ao qual está conectado, repetindo esse procedimento caso não consiga adquirir um endereço na rede;
2. Na sequência, as portas de E/S que serão utilizadas são configuradas para o modo *INPUT* ou entrada, para receber os sinais dos sensores;
3. A partir de então, é verificado o estado de cada sensor:
 - A. Sensor fechado: estado padrão do sistema, significa que a porta ou janela está fechada;
 - B. Sensor aberto: significa que a porta ou janela está aberta:
 - I. É verificado se o sensor já está cadastrado na lista de sensores ativos:
 - a) Se este sensor for encontrado na lista, é passado para verificar o próximo sensor;
 - b) Se não for encontrado o sensor, é realizado o cadastro do mesmo (número da porta utilizada, e seu estado igual a aberto) e passado para verificar o próximo sensor;

Ao final da verificação de todos os sensores, é iniciada uma análise da lista de sensores ativados a partir do primeiro sensor cadastrado.

4. É procurado pelo primeiro sensor que possua um dos estados listados abaixo, exceto o estado aguardando que necessita ter o estado atual do sensor igual a fechado:
 - A. Aberto: muda o estado do sensor cadastrado na lista para “enviar um *tweet*”;
 - B. Enviar um *tweet*: envia um tweet para a conta cadastrada informando qual o sensor que foi ativado e passando seu estado para “enviar para o site”;

- C. Enviar para o site: envia uma mensagem para o sistema de monitoramento residencial informando qual o sensor que foi ativado e logo após, muda o estado do sensor cadastrado na lista para aguardando;
- D. Aguardando: remove o sensor da lista, caso o estado atual do sensor for igual a fechado;

Ao término da parte quatro, volta-se para a parte três, verificando o estado de todos os sensores. Pode-se destacar que a etapa quatro trabalha somente com um sensor cadastrado na lista por vez, isto é necessário para ter um melhor controle dos sensores do sistema.

Estes métodos de programação foram utilizados para desenvolver um algoritmo que ocupe o mínimo de espaço possível, já que o Arduino Mega 1280 possui apenas 128Kb de memória. O algoritmo desenvolvido faz com que o Arduino assuma a posição de cliente no sistema, pois quando ocorre algum evento, ele se conecta ao sistema Web para reportar informações sobre o evento e também enviar, automaticamente, um *tweet* para a conta configurada no *Web site* <www.twitter.com>, avisando qual sensor foi ativado.

4.2 Sistema Web

O sistema Web tem por finalidade armazenar as informações enviadas pelo Arduino, bem como algumas informações referentes ao usuário (endereço, *e-mail*, telefone, *twitter*) e aos sensores utilizados. Também é responsável por permitir um acesso rápido e prático ao sistema de monitoramento residencial em qualquer aparelho que possua acesso à Internet.

4.2.1 Modelagem do Banco de Dados

Para realizar o desenvolvimento do banco de dados do sistema de monitoramento, foi feito um levantamento dos requisitos básicos que deveriam constar no modelo conceitual do banco de dados. Tais requisitos estão elencados abaixo:

- ✓ Um cliente poderá ter mais de uma residência monitorada;
- ✓ Um Arduino irá realizar o monitoramento de somente uma residência;
- ✓ Cada Arduino deverá possuir um status (*ON* ou *OFF* ou *ALERTA*);
- ✓ Cada sensor deve estar ligado em um único Arduino;
- ✓ Deve-se saber o horário (data e hora) do acontecimento de cada evento, assim como o sensor que originou;
- ✓ A localização e o tipo de cada sensor instalado;
- ✓ A localização, a descrição e as coordenadas geográficas de cada Arduino.

Após o levantamento de todos os requisitos básicos, foi realizado o projeto conceitual do banco de dados, conforme mostrado na *Figura 13* abaixo:

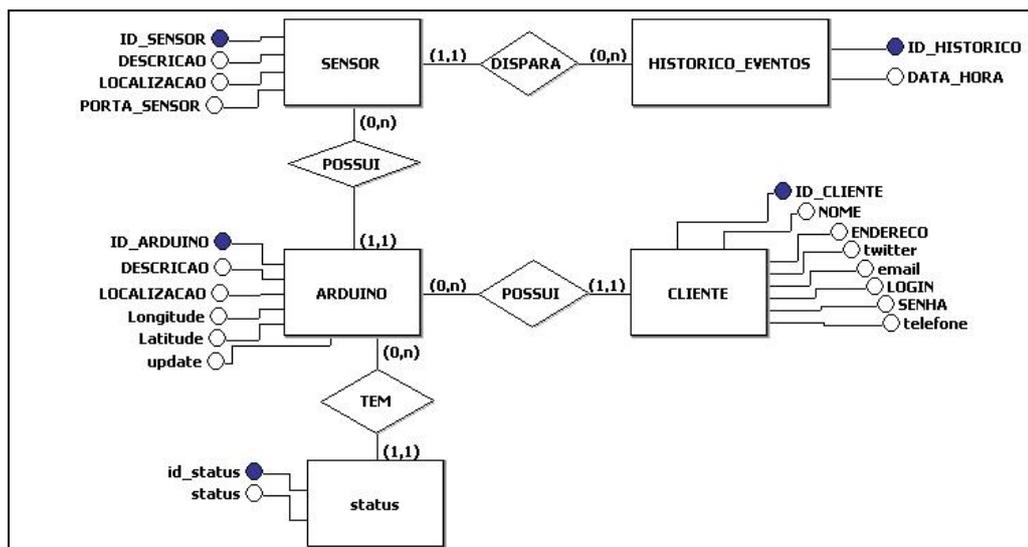


Figura 13 - Projeto conceitual do banco de dados.

Realizado o projeto conceitual e definidas as cardinalidades do banco, foi realizado o projeto lógico para definir os tipos das variáveis e as chaves estrangeiras. Esta etapa pode ser visualizada na *Figura 14*.

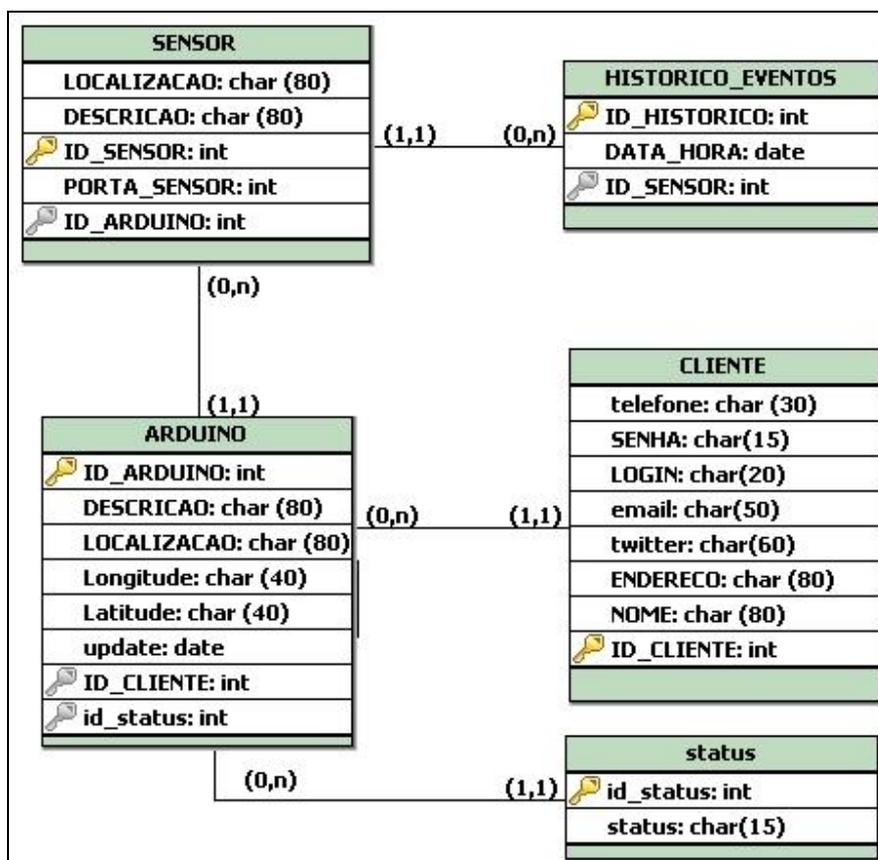


Figura 14 - Modelo entidade relacionamento.

Concluído o projeto lógico, foi realizada a implementação do projeto físico utilizando o SGBD MySQL e criando as cinco tabelas, que são brevemente descritas abaixo:

A tabela “cliente” é responsável por armazenar as informações (nome, endereço, *login*, senha) do usuário;

A tabela “arduino” é o local em que são dispostos os dados de cada Arduino, pois conforme a modelagem do banco, um usuário poderá ter mais de uma residência cadastradas. Entre as diversas informações cadastradas nesta tabela,

está a “descrição” (ex.: casa de praia), endereço, coordenadas geográficas para o posicionamento no mapa disponibilizado pela aplicação. Também nesta tabela, encontra-se o real estado do sistema, onde o “*id_status*” poderá assumir os seguintes valores conforme a tabela “status”:

- 1: dispositivo em modo *ON*;
- 2 : dispositivo em modo *OFF*;
- 3: dispositivo em modo ALERTA;

A coluna “*up*” da tabela “arduino” é atualizada em intervalos determinados de tempo, nos quais o Arduino envia um sinal para o sistema de monitoramento, informando que ainda está em funcionamento. Caso o sistema não receba esta confirmação, o estado do Arduino será alterado para *OFF*.

Na tabela “sensor”, cada sensor terá armazenado seu tipo (*magnético*, de movimento, pressão), local onde foi instalado (janela da cozinha, porta da garagem), o respectivo arduino ao qual pertence e em qual porta do arduino está associado. Cada sensor poderá estar cadastrado a somente um Arduino.

A tabela “histórico de eventos” é responsável por armazenar todos os eventos ocorridos com os sensores, registrando a data e hora do evento, assim como o sensor que o gerou.

4.2.2 Desenvolvimento do sistema Web

No sistema Web, o usuário poderá realizar algumas interações no sistema, tais como:

- Inserir, alterar e excluir sensores;
- Inserir, alterar e excluir Arduinos;
- Acessar os eventos ocorridos nas residências cadastradas pelo usuário;
- Acessar um mapa com as coordenadas das residências e a atual situação do sistema (*ON*, *OFF*, ALERTA).

Serão demonstradas algumas telas do sistema a fim de ilustrar os recursos disponíveis.

Na *Figura 15* é apresentada a tela inicial do sistema Web, onde é necessário utilizar usuário e senha para ter acesso aos recursos da aplicação. Este é um requisito básico de segurança para evitar o acesso de pessoas não autorizadas ao sistema.

Sistema De Monitoramento Residencial

Segunda-feira, 25 de Junho de 2012

Login

Senha

Figura 15 - Acesso ao Sistema Web.

Com a confirmação das credenciais, o usuário pode ter acesso ao menu da aplicação, em que é possível acessar o histórico dos eventos ocorridos, informando a data e a hora do evento ocorrido, como também qual sensor originou o evento. Essa tela pode ser vista na *Figura 16*.

Sistema De Monitoramento Residencial

Segunda-feira, 25 de Junho de 2012

Evento	Local Arduino	Horario	Localização do Sensor
1319	casa principal	2012-05-25 08:06:13	JANELA-QUAR
1318	casa principal	2012-05-25 07:44:36	JANELA-QUAR
1320	casa principal	2012-05-25 08:07:41	JANELA-QUAR
1321	casa principal	2012-05-25 08:42:24	JANELA-QUAR
1323	casa principal	2012-05-25 16:33:40	JANELA-QUAR
1324	casa principal	2012-05-25 16:37:04	JANELA-QUAR
1326	casa principal	2012-05-25 16:38:08	JANELA-QUAR
1327	casa principal	2012-05-25 16:57:43	JANELA-QUAR

Figura 16 - Histórico de eventos ocorridos.

O sistema também conta com a possibilidade de inserir, alterar e excluir os Arduinos cadastrados em sua conta. A Figura 17 mostra as telas de manipulação dos Arduinos.

Sistema De Monitoramento Residencial

Segunda-feira, 25 de Junho de 2012

ARDUINO

Cadastrar

Editar

SENSORES

Cadastrar

Editar

EVENTOS

MAPA

SAIR

Adicionar Novos Arduinos

Descrição (ex.: casa de praia)	<input type="text"/>
Localização:(endereço)	<input type="text"/>
Latitude:	<input type="text"/>
Longitude	<input type="text"/>
<input type="button" value="Adicionar"/> <input type="button" value="Limpar"/>	

Sistema De Monitoramento Residencial

Segunda-feira, 25 de Junho de 2012

ARDUINO

Cadastrar

Editar

SENSORES

Cadastrar

Editar

EVENTOS

MAPA

SAIR

ID Arduino	Descrição	Localização	Opção
1	casa principal	Rs 149, km 144.	Alterar Excluir
3	Casa das amoras	Santa maria	Alterar Excluir
Total de Arduinos encontrados: 2			

Figura 17. - Cadastro de Arduino.

Para melhor visualização da atual situação do sistema de monitoramento residencial, foi utilizada uma aplicação disponibilizada pelo *Google*, onde é gerado um mapa utilizando as coordenadas geográficas cadastradas no sistema e marcando com um balão a posição da residência. Isto pode ser visualizado na *Figura 18*.

Este balão poderá assumir três cores: vermelho, quando sistema está desligado; verde quando sistema está funcionando sem evento ocorrido desde que foi ligado; e amarelo quando ocorrer algum evento.



Figura 18 - Mapa de monitoramento.

Ao clicar sobre o balão são mostradas algumas informações do Arduino em questão, como endereço, tempo de funcionamento e em caso de alerta, qual foi o sensor ou sensores que foram ativados (*Figura 19*).



Figura 19 - Informações disponíveis em cada balão.

Ainda é possível cadastrar, alterar e excluir os sensores, como pode ser visualizado na Figura 20.

Sistema De Monitoramento Residencial

Segunda-feira, 25 de Junho de 2012

- ARDUINO
- SENSORES
- Cadastrar
- Editar
- EVENTOS
- MAPA
- SAIR

Adicionar Novos Sensores

Descrição (tipo do sensor)	<input type="text"/>
Localização: (ex.: portão garagem)	<input type="text"/>
Porta do Arduino:	<input type="text"/>
Arduino	<input type="text" value="casa principal"/>
<input type="button" value="Adicionar"/> <input type="button" value="Limpar"/>	

Sistema De Monitoramento Residencial

Segunda-feira, 25 de Junho de 2012

- ARDUINO
- SENSORES
- Cadastrar
- Editar
- EVENTOS
- MAPA
- SAIR

ID Sensor	Arduino	Descrição	Localização	Opção
28	casa principal	reed switch	JANELA-QUAR	Alterar Excluir
30	casa principal	Sensor de preseça	cozinha	Alterar Excluir
Total de Sensores encontrados: 2				

Figura 20 - Cadastro dos sensores

4.3 Sistema de monitoramento residencial – SiMRe

Após a implementação do fluxograma responsável por tomar as decisões na plataforma Arduino e o desenvolvimento do sistema Web, obteve-se o sistema de monitoramento residencial - SiMRe, que é mostrado na *Figura 21*:

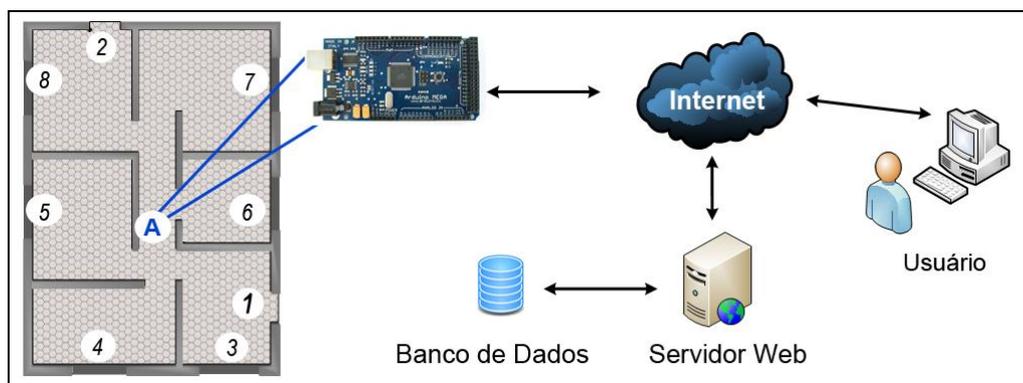


Figura 21 - Sistema de Monitoramento Residencial – SiMRe.

O SiMRe é composto das seguintes partes:

- ✓ Arduino: responsável por monitorar os sensores instalados na residência;
- ✓ Sensores: responsáveis por monitorar as portas e janelas da residência;
- ✓ *Ethershield*: realiza a interface de comunicação entre o Arduino e o sistema Web;
- ✓ Sistema Web: realiza a interface entre o usuário, Arduino e sensores;
- ✓ Mapa: permite a visualização dos Arduinos monitorados;
- ✓ Twitter: possibilita ao usuário receber notificações em tempo real enviadas pelo SiMRe, através do envio de um *tweet*, para a conta configurada caso ocorra algum evento na residência.

Abaixo podemos visualizar a maquete do projeto proposto (*Figura 22*), simulando uma casa com uma porta e uma janela sendo monitoradas por sensores magnéticos. Esta maquete tem por finalidade materializar o projeto, para que se possa ver seu real funcionamento.



Figura 22 - Maquete do projeto proposto.

Os custos para realizar a implementação do projeto proposto pode ser analisado no quadro abaixo:

Dispositivo	Preço
Arduino Mega 1280	R\$ 110,00
<i>Ethershield</i> ENC28J60	R\$ 50,00
Sensores <i>Magnéticos</i>	R\$ 5,00
Outros componentes (fios, estanho, resistores)	R\$ 20,00
Total	R\$ 185,00

Quadro 3 - Custos do projeto.

O custo do projeto pode variar, dependendo da quantidade e o tipo de sensores utilizados, além da utilização dos fios para interligar os diversos sensores ao Arduino que para o projeto proposto foi desconsiderado.

5 CONSIDERAÇÕES FINAIS

Este trabalho apresentou a elaboração de um protótipo de um sistema de monitoramento residencial utilizando a plataforma Arduino.

O trabalho surgiu após a constatação de que a segurança residencial tem se tornado algo popular e com muitos gastos no mercado - veja-se as notícias diárias de assaltos à casas de pessoas de todas as classes: baixa, média e alta. Dessa forma, pensando especificamente em atender o mercado voltado para as famílias de classes baixa e média, que não tem condições de arcar com os custos de câmaras de vigilância, segurança particular 24 horas, propôs-se a construção desse sistema autônomo de segurança residencial de baixo custo.

Com o sistema proposto neste trabalho, o usuário pode visualizar à distância (do trabalho, da rua etc.) e em “tempo real” como está a situação em sua residência. Todas as informações são acessadas pelo usuário através da Internet. Caso algum sensor detecte a presença de algo, o usuário saberá exatamente em qual cômodo da casa ocorreu o fato, e em tempo real. O sistema implementado prevê também o armazenamento de informações. Assim, pode-se fazer um histórico das atividades/anomalias ocorridas na residência durante um determinado espaço de tempo. O sistema também possibilita a existência de várias casas cadastradas para um mesmo usuário, assim o usuário saberá se ocorreu algum arrombamento; em qual das casas; e em que cômodo da casa foi o fato.

Os testes realizados no protótipo da casa, desenvolvido no projeto, mostraram que o sistema apresentou-se estável, tornado-se passível para ser implementado em uma residência. Seu custo também se mostrou acessível.

A plataforma Arduino utilizada no sistema proposto é totalmente *open source* tanto em hardware, quanto no software para desenvolver os algoritmos, o que garante um baixo custo na aquisição do equipamento. Devido a estas características, é possível que outras pessoas desenvolvam novos dispositivos de *hardware* que interajam com o Arduino o que contribui ainda mais para baixar o custo do sistema.

O algoritmo proposto por este projeto também poderá ser customizado conforme as necessidades de cada usuário, tornando um sistema flexível, uma vez que o código fonte é disponibilizado como Apêndice A.

Como pontos negativos deste sistema, pode-se citar a inexistência de câmeras - o que tornaria o projeto mais caro, mas contribuiria na segurança e na visualização, em tempo real, dos fatos que estão ocorrendo na residência.

Por fim, como sugestões de trabalhos futuros:

- a) Com relação ao algoritmo desenvolvido para o Arduino, sugere-se implementar um mecanismo para ativar/desativar o sistema pela Internet; e um mecanismo para remover os fios que ligam o Arduino aos sensores, substituindo-os por uma rede de sensores sem fio;
- b) Com relação ao sistema Web, sugere-se desenvolver um sistema alternativo, próprio para o acesso por dispositivos móveis, como celulares.

REFERÊNCIAS

ABOUT the Apache HTTP Server Project... **Apache**. [S.l.] . Disponível em: <http://httpd.apache.org/ABOUT_APACHE.html>. Acesso em: 04 jun. 2012.

ALECRIM, Emerson. **Banco de dados MySQL e PostgreSQL**. [S.l.], 2008. Disponível em: < <http://www.infowester.com/postgremysql.php>>. Acesso em: 21 jun. 2012.

ARDUINO – home page. **Arduino**. [S.l.]. Disponível em: <<http://www.arduino.cc>>. Acesso em: 27 abr. 2012.

ARDUINO Mega. **Arduino**. [S.l.]. Disponível em: <<http://arduino.cc/en/Main/ArduinoBoardMega>>. Acesso em: 28 abr. 2012.

ARDUINO Mega 2560. **Arduino**. [S.l.]. Disponível em: < <http://arduino.cc/en/Main/ArduinoBoardMega2560>>. Acesso em: 28 abr. 2012.

ARDUINO Uno. **Arduino**. [S.l.]. Disponível em: <<http://arduino.cc/en/Main/ArduinoBoardUno>>. Acesso em: 28 abr. 2012.

BARRETTA, Rodrigo et al. 3S: Uma Solução de Sensoriamento Segura para Ambientes Domésticos. In: SIMPÓSIO BRASILEIRO EM SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS. 9., 2009, Campinas-SP. **Anais...** Campinas: [s.n.], 2009. p. 301-310. Disponível em: <http://labcom.inf.ufrgs.br/ceseg/anais/2009/Anais_SBSeg_2009.pdf>. Acesso em: 30 abr, 2012.

BAUMANN, Daniel. **Protótipo de um sistema de segurança residencial com Linux embarcado**. 2008. 49 f. Trabalho de Conclusão de Curso (Bacharel em Ciência da Computação) – Curso de Ciência da Computação, Universidade Regional de Blumenau, Blumenau, 2008. Disponível em: <www.bc.furb.br/docs/MO/2008/330370_1_1.pdf>. Acesso em: 30 abr. 2012.

CARVALHO, Mauricio Feo Pereira Rivello de. Automação e controle residencial via internet utilizando arduino. In: SEMANA DE EXTENSÃO, 1., Rio de Janeiro. **Anais...** Rio de Janeiro: [online], 2011. Disponível em:

<http://portal.cefetrij.br/files/extensao/outros/livro_sem_ext_2011.pdf#page=34>. Acesso em: 20 jun. 2012.

CRIMES motivam procura por sistemas de segurança. **Jornal Da Manhã**, Minas Gerais, 27 fev. 2012. Disponível em: <<http://jmonline.com.br/novo/?noticias,1,GERAL,58041>>. Acesso em: 08 mar. 2012.

ELMASRI, Ramez, **Sistemas de banco de dados**. 4. ed. São Paulo: Pearson Addison Wesley, 2005.

HACKENHAAR, Jonathan; CARDOSO, Tatiana. **Um comparativo entre PHP e JSP: definindo a melhor aplicação para o desenvolvimento de projetos web**. [S.l.], Revista iTEC – Vol. I, Nº 1, 2010.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. Características da vitimação e do acesso à justiça no Brasil: 47,2% das pessoas não se sentem seguras na cidade em que moram. [S.l.], 15 dez. 2010. Disponível em: <http://www.ibge.gov.br/home/presidencia/noticias/noticia_impresao.php?id_noticia=1786>. Acesso em: 20 fev. 2012.

JANUARY 2012 Web Server Survey. **Netcraft**. [S.l.], 03 jan. 2012. Disponível em: <<http://news.netcraft.com/archives/2012/01/03/january-2012-web-server-survey.html>>. Acesso em 04 jun. 2012.

KLABUNDE, Fabrício. **Software para monitoramento de servidores web Apache**. 2007. 28 p. Trabalho de Conclusão de Curso (Bacharel em Ciências da computação) – Universidade Regional de Blumenau.

LIMA, Charles Borges de; SCHWARZ, Leandro. Kit didático para trabalho com os microcontroladores AVR – Kit ATMEGA++. **Revista ilha digital**, [online], v. 1, p. 93-99, 2009. Disponível em: <<http://ilhadigital.florianopolis.ifsc.edu.br/index.php/ilhadigital/article/view/777>>. Acesso em: 20 jun. 2012.

MARCHETTE, Murilo Marchette; NUNES, Reginaldo Barbosa. Redes sem fio aplicadas à automação e ao monitoramento residencial. In: JORNADA DE INICIAÇÃO CIENTÍFICA, DESENVOLVIMENTO TECNOLÓGICO E INOVAÇÃO, 4., 2011, Vitória - ES. **Anais...** Vitória: ed. do IFES, 2011. Não paginado. Disponível em: <http://pse.ifes.edu.br/prppg/pesquisa/jornadas/jornada_2010_2011/anais/045_anais_do_evento_arquivos/..%5CT2419.pdf>. Acesso em: 30 abr. 2012.

MARTINS, N. A. **Sistemas Microcontrolados: Uma Abordagem com o Microcontrolador PIC 16F84**. São Paulo: Novatec, 2005. Disponível em: <<http://www.livrariacultura.com.br/imagem/capitulo/3173664.pdf>>. Acesso em: 20 jun. 2012.

MONARO, Renato Machado. **Sistema de aquisição de dados para um relé de proteção digital de baixo custo**. 2007. Trabalho de Conclusão de Curso (Bacharel em Engenharia Elétrica com ênfase em sistemas de energia e automação) – Universidade de São Paulo.

PEREIRA, Daniela. Brasileiros investem em segurança residencial. **Tribuna da Bahia**. Bahia, 25 fev. 2012. Disponível em: <<http://www.tribunadabahia.com.br/news.php?idAtual=107060>>. Acesso em 01 mar. 2012.

PEREIRA, Monique Madeira. **Estudo e desenvolvimento de uma aplicação para o monitoramento remoto de sistemas de segurança doméstica, utilizando dispositivos móveis**. 2009. 82 f. Trabalho de Conclusão de Curso (Bacharel em Ciência da Computação) – Curso de Ciência da Computação, Universidade do Estado de Santa Catarina, Joinville, 2009. Disponível em: <<http://www.pergamum.udesc.br/dados-bu/000000/000000000000D/00000D62.pdf>>. Acesso em: 30 abr. 2012.

PESQUISA revela os novos hábitos de assistir TV dos brasileiros. **Olhar Digital**, [S.l.], 13 fev. 2012. Disponível em: <<http://olhardigital.uol.com.br/produtos/mobilidade/noticias/pesquisa-revela-os-novos-habitos-de-assistir-tv-dos-brasileiros>>. Acesso em: 08 mar 2012.

SILVA, Davidson Felipe da. **Sistema de comunicação Bluetooth utilizando microcontrolador**. 2009. 17 f. Trabalho de Conclusão de Curso (Bacharel em Engenharia da Computação) – Curso de Engenharia da Computação, Escola Politécnica de Pernambuco – Universidade de Pernambuco.

SOBRE o postgresql. **PostgreSQL**. [S.l.]. Disponível em: <<http://www.postgresql.org.br/sobre>>. Acesso em: 21 jun. 2012.

TANENBAUM, A. **Redes de Computadores**. 4. ed. [S.l.]: Editora Campus. 2003.

APÊNDICE

Apêndice A – Algoritmo desenvolvido para o Arduino

```
#include <EtherCard.h>
#define LED_STATUS 13
struct Struct{ //struct da lista sensores ativados;
  int sensor;
  int estado;
  struct Struct* prox;
};
struct Struct *lista = NULL;

enum ENUM_SENSORES {
  low, high, env_tweet, env_site, aguardando}; // estados do sensor na lista;
static int TAM = 2; // tamanho do vetor;
int Sensores[2] = {
  30,28}; //portas utilizadas;

static byte mymac[] = {
  0x74,0x69,0x69,0x2D,0x30,0x31 };
byte Ethernet::buffer[700];
static uint32_t timer, timerS;
char website[] PROGMEM = "www.monitoramentoresidencial.site11.com";
char tweet[] PROGMEM = "api.supertweet.net";
#define KEY "bWFyY2Vsb2ZheGluYWw6YXJkdWlubw=="
Stash stash;
//-----
static void enviarParaTwitter (int sensor) {
  byte sd = stash.create();
  stash.print("status= Warning ID_SENSOR =");
  stash.println(sensor);
  stash.save();
  Stash::prepare(PSTR("POST /1/statuses/update.xml HTTP/1.1" "\r\n"
    "Host: $F" "\r\n"
    "Authorization: Basic $F" "\r\n"
    "User-Agent: Arduino Residencial" "\r\n"
    "Content-Length: $D" "\r\n"
    "Content-Type: application/x-www-form-urlencoded" "\r\n"
    "\r\n"
    "$H"),
  tweet, PSTR(KEY), stash.size(), sd);
  ether.tcpSend();
}
//-----
static void enviarParaSite(int sensor){ //envia para o sistema qual sensor
  byte sd = stash.create(); //foi ativado;
  stash.print("ID_SENSOR= ");
  stash.println(sensor);
  stash.save();
  Stash::prepare(PSTR("POST /hidden.php HTTP/1.1" "\r\n"
    "Host: $F" "\r\n"
    "Authorization: Basic $website" "\r\n"
    "User-Agent: Arduino Residencial" "\r\n"
    "Content-Length: $D" "\r\n"
    "Content-Type: application/x-www-form-urlencoded" "\r\n"
    "\r\n"

```

```

    "$H"),
    website, NULL, stash.size(), sd);
    ether.tcpSend();
}
//-----
static void estouVivo(){ //envia notificações para o sistema dizendo
    byte sd = stash.create(); // que o arduino ainda esta ligado;
    stash.print("ID_ARDUINO= 1");
    stash.save();
    Stash::prepare(PSTR("POST /hidden.php HTTP/1.1" "\r\n"
        "Host: $F" "\r\n"
        "Authorization: Basic $website" "\r\n"
        "User-Agent: Arduino Residencial" "\r\n"
        "Content-Length: $D" "\r\n"
        "Content-Type: application/x-www-form-urlencoded" "\r\n"
        "\r\n"
        "$H"),
        website, NULL, stash.size(), sd);
    ether.tcpSend();
}
//-----INSERIR-LISTA-----
struct Struct *inserir(struct Struct *lista, int valor){ //insere um sensor
struct Struct *novo; //na lista;
struct Struct *aux;

    novo = (struct Struct *) malloc(sizeof(struct Struct)); //alocar memória;
    novo->sensor = valor;
    novo->prox = NULL;
    novo->estado = low;
    Serial.print("Sensor [");
    Serial.print(novo->sensor);
    Serial.print("] ATIVADO! \n");
    if(lista == NULL){
        return novo;
    }
    else{
        for(aux = lista ; aux->prox ; aux = aux->prox);
        aux->prox = novo;
        return lista;
    }
}
//-----IMPRIMIR LISTA-----
void imprimir(struct Struct *lista){
    int i = 0;
    if(lista != NULL)
        for(struct Struct *aux=lista; aux;aux= aux->prox){
            Serial.print("Posicao[");
            Serial.print(i);
            Serial.print("] =");
            Serial.print(aux->sensor);
            Serial.print(" Estado = ");
            Serial.println(aux->estado);
            i++;
        }
    else
        Serial.println("LISTA VAZIA");
}
//-----

```

```

struct Struct * remover(struct Struct *lista, int sensor){ //remover sensor
struct Struct *aux, *p;
                                //da lista;

Serial.print("Sensor [");
Serial.print(sensor);
Serial.print("] DESATIVADO! \n");
if (lista->sensor == sensor){ //se o sensor a ser removido é o primeiro
    aux = lista->prox;        //da lista;
    free(lista);
    return aux;
}
else{
    for(aux = lista; aux->sensor != sensor && aux ; aux= aux->prox)
        p = aux;
    if(p->prox->prox != NULL) //verifica se há outro sensor após o
        p->prox = p->prox->prox;// sensor corrente;
    else //Nao ha nenhum elemento após o sensor corrente;
        p->prox = NULL;
    free(aux);
    return lista;
}
}
//-----
int verificaLista(int sensor, struct Struct *lista){
struct Struct *aux;
for(aux = lista; aux; aux= aux->prox) //busca toda a lista procurando
    if(aux->sensor == sensor) //pelo sensor ativado;
        return 0; // foi encontrado o sensor na lista
return 1; //não foi encontrado o sensor na lista
}
//-----
struct Struct *verificaSensores(struct Struct *lista){ // verifica todos
int i, aux = 0;
                                //os sensores;
for (i=0; i<TAM; i++)
    if( digitalRead(Sensores[i]) == LOW){//procurar por um sensor ativado;
        aux = verificaLista(Sensores[i], lista);//sensor já está na lista?
        if(aux){ // 1 - NÃO; 0-SIM
            lista = inserir(lista, Sensores[i]);
        }
    }
}
return lista;
}
//=====
static void gotPinged (byte* ptr) {
    ether.printlp(">>> ping from: ", ptr);
}
void setup () { //Ã‰ executado somente na inicialização do Arduino
int i=0;
Serial.begin(57600); //velocidade da comunicação serial
Serial.println("\n[Cliente - Arduino]");
//configuração DHCP
if (ether.begin(sizeof Ethernet::buffer, mymac) == 0)
    Serial.println( "Failed to access Ethernet controller");
if (!ether.dhcpSetup())
    Serial.println("DHCP failed");

    ether.printlp("IP: ", ether.myip);
    ether.printlp("GW: ", ether.gwip);
}

```

```

ether.println("DNS: ", ether.dnsip);
if (!ether.dnsLookup(website))
  Serial.println("DNS failed");
ether.println("SRV: ", ether.hisip);

pinMode(LED_STATUS, OUTPUT);

for (i=0; i<TAM; i++) //Seta os sensores para INPUT
  pinMode(Sensores[i], INPUT);
Serial.println(">>agora vai<<");
estouVivo();
}
//=====
void loop () { //Código executa em loop infinitamente
  int var = 0;
  struct Struct *aux, *teste;
  word len = ether.packetReceive();
  word pos = ether.packetLoop(len);

  //resposta ping
  if (len > 0 && ether.packetLoopcmpCheckReply(ether.hisip)) {
    Serial.print(" ");
    Serial.print((micros() - timer) * 0.001, 3);
    Serial.println(" ms");
  }
  ///

  aux = lista;
  lista = verificaSensores(lista); //verifica se algum sensor foi ativado
  ether.packetLoop(ether.packetReceive());

  if (lista){ // Existe a lista?
    while (aux->estado == aguardando && digitalRead(aux->sensor) == LOW)
      if (aux->prox != NULL)
        aux = aux->prox;
      else
        break;

    switch (aux->estado){
    case low: // sensor ativado
      aux->estado = env_tweet;
      Serial.println("-----");
      Serial.print("Sensor [");
      Serial.print(aux->sensor);
      Serial.print("] Mudanca de estado: aberto para enviar tweet \n");
      break;
    case env_tweet: // enviar um tweet para o twitter;
      if (millis() > timerS) {
        timerS = millis() + 1500;
        enviarParaTwitter(aux->sensor);
        aux->estado = env_site; //mudança de estado;
        Serial.print("\n tweet=");
        Serial.println(aux->sensor);
        Serial.print("Sensor [");
        Serial.print(aux->sensor);
        Serial.print("] Mudanca de estado: enviar tweet para enviar site \n");
      }
    }
  }
}

```

```

break;
case env_site: //enviar mensagem para o sistema
  if (millis() > timerS) {
    timerS = millis() + 1000;
    enviarParaSite(aux->sensor);
    aux->estado = aguardando;//mudança de estado;
    Serial.print("site = ");
    Serial.println(aux->sensor);
    Serial.print("Sensor [");
    Serial.print(aux->sensor);
    Serial.print("] Mudanca de estado: enviar site para final \n");
    Serial.println("----- ");
  }
  break;
case aguardando: //aguarda o sensor voltar ao estado inicial (desativado);
  if(digitalRead(aux->sensor) == HIGH){
    lista = remover(lista, aux->sensor);//remover sensor da lista;
  }
  break;
}
}
//manda sinal de vida a cada 2 minutos 120000
if (millis() > timer) {
  timer = millis() + 120000;
  estouVivo(); //envia mensagem para o sistema, avisando
  Serial.print("ESTOU VIVO A= ");//que o Arduino está vivo;
  Serial.print(timer/1000);
  Serial.println(" segundos ");
}
}

```